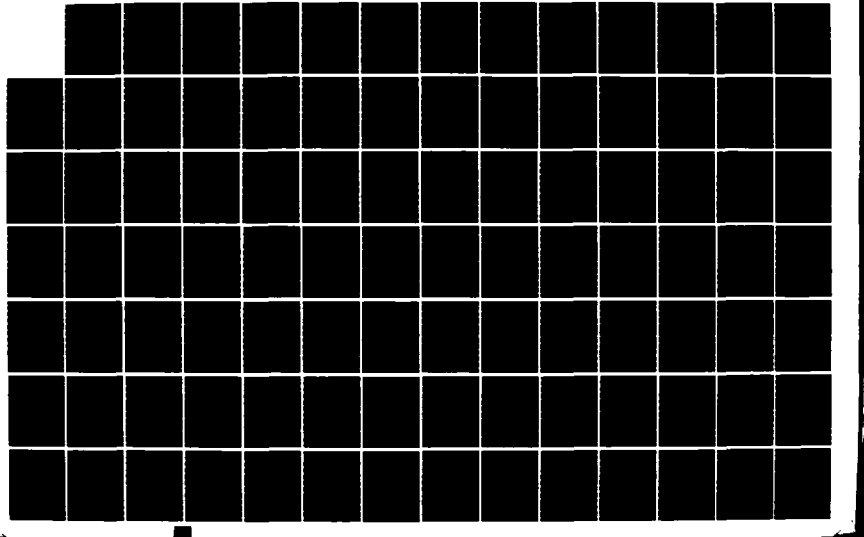
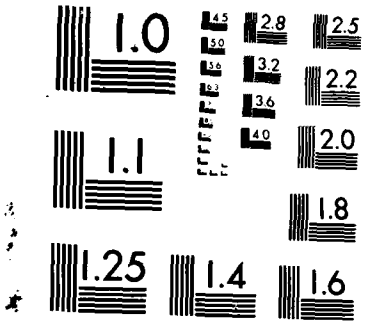
AD-A173 903 AN AUTOMATED SYSTEM FOR THE CONTROL OF AND DATA ACQUISITION FROM MULTIPHO..(U) ARMY BALLISTIC RESEARCH LAB ABERDEEN PROVING GROUND MD M A DEWILDE SEP 86 1/2

UNCLASSIFIED BRL-TR-2759 F/G 9/2 NL





MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

(12) (8)

AD

US ARMY MATERIEL COMMAND

TECHNICAL REPORT BRL-TR-2759

AD-A173 903

AN AUTOMATED SYSTEM FOR THE CONTROL OF, AND DATA ACQUISITION FROM MULTIPHOTON IONIZATION AND FLUORESCENCE LIFETIME MEASUREMENTS

Mark A. DeWilde

DTIC FILE COPY

NOV 1 2 1986

September 1986

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

US ARMY BALLISTIC RESEARCH LABORATORY
ABERDEEN PROVING GROUND, MARYLAND

Destroy this report when it is no longer needed.
Do not return it to the originator.

Additional copies of this report may be obtained from the National Technical Information Service, U. S. Department of Commerce, Springfield, Virginia 22161.

The findings in this report are not to be construed as an official Department of the Army position, unless so designated by other authorized documents.

The use of trade names or manufacturers' names in this report does not constitute indorsement of any commercial product.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER Technical Report BRL-TR-2759	2. GOVT ACCESSION NO. AD-A173903	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) AN AUTOMATED SYSTEM FOR THE CONTROL OF, AND DATA ACQUISITION FROM MULTIPHOTON IONIZATION AND FLUORESCENCE LIFETIME MEASUREMENTS		5. TYPE OF REPORT & PERIOD COVERED Final Report 6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Mark A. DeWilde		8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS U.S. Army Ballistic Research Laboratory ATTN: SLCBR-IB Aberdeen Proving Ground, MD 21005-5066		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS 1L161102AH43
11. CONTROLLING OFFICE NAME AND ADDRESS U.S. Army Ballistic Research Laboratory ATTN: SLCBR-DD-T Aberdeen Proving Ground, MD 21005-5066		12. REPORT DATE September 1986 13. NUMBER OF PAGES 140
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)		15. SECURITY CLASS. (of this report) UNCLASSIFIED 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE N/A

16. DISTRIBUTION STATEMENT (of this Report)

Approved for Public Release; Distribution Unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Multiphoton Ionization Spectroscopy
Fluorescence Lifetime
IEEE-488 Interface
Dec PDP-11 Computer

20. ABSTRACT (Continue on reverse side if necessary and identify by block number) meg, jmk

An automated system for data acquisition and control via IEEE-488 interface to a general purpose laboratory computer is described. Samples of data taken with the system on a multiphoton ionization spectroscopy experiment are shown, and all necessary additional control systems described. It is believed by the author that the IEEE-488 control system developed for this work represents a superior interface for the Dec PDP-11 computer than that available from the manufacturer. The software packages presented can be easily altered to support a wide range of IEEE-488 bus controllable devices.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

TABLE OF CONTENTS

		Page
	LIST OF FIGURES	5
I.	INTRODUCTION	7
II.	THE HARDWARE INTERFACES	7
III.	THE HARDWARE DRIVER ROUTINES	16
IV.	THE USER SOFTWARE SYSTEMS	35
V.	THE ACCESSORY A/I DATA FILE PLOTTING PROGRAM	104
VI.	CLOSING COMMENTS	111
	ACKNOWLEDGEMENTS	116
	REFERENCES	117
	APPENDIX	119
	DISTRIBUTION LIST	137

QUALITY INSPECTED 4

A-1

LIST OF FIGURES

Figure		Page
1	Transient Digitizer Connections	8
2	TTL Logic and Bussing	9
3	BLIS to IEEE-488 Converter Box	11
4	Auxiliary Control Connections	14
5	The Stepper Motor Connections	15
6	The Architecture of the IEEE-488 Control System	20
7	The Architecture of the User Level Programs	36
8	Single Buffer Data Plot Sample	112
9	Double Buffer Data Plot Sample	113
10	Triple Buffer Data Plot Sample	114
11	A/I Buffer Plot from Auxiliary Plotting Program	115

I. INTRODUCTION

In a previous report,[1] a general purpose laboratory data acquisition and control system was described. The systems addressed in this work build upon that system, to provide the means to take fast transient pulse measurements in laboratory optical experiments. In the case of the fluorescence lifetime measurements, a chemical species is excited with a short (20 nanosecond) pulse of laser light at various wavelengths, and the radiative lifetime is measured by observing the fluorescence with a fast photomultiplier tube. The voltage pulse is amplified and sent to a Tektronix model 7912AD transient digitizer, which digitizes and stores the signal. In order to use this data, a computer is required to control the 7912AD and to take the stored waveform and make the data available to the user in numerical form. The 7912AD uses the IEEE-488 standard interface as its means of communications with the host computer. For complete information on the 7912AD, the reader is referred to Reference 2.

The multiphoton ionization experiments consist of using a single pulse, or two simultaneous pulses, of laser light to pump atomic species through excited levels to ionization. The energy required for ionization is supplied in these cases by more than one photon of light, unlike the lifetime work. The signals to be detected fall into two categories: light pulses, as in the fluorescence work, and electrical signals from direct ion pickup. Both signals again take the form of very fast transient electrical pulses and are acquired via the 7912AD digitizer. In addition, due to the nature of the measurements, other features of the experimental setup were automated. The scanning of laser wavelength, the opening and closing of a shutter, the movement of a burner system, and the switching of a high voltage power supply were automated in order to do various types of experiments, and to subtract noise from the useful signal.

The remainder of this report will be devoted to the description of the hardware and software systems that were developed in order to fill the needs of these two experimental studies, and to the philosophy of the data acquisition and control programs. These adhere to most of the rules that are laid down in Reference 3, departing only where pragmatism demanded. As the hardware and software is built upon the systems described in Reference 1, the reader is assumed to be familiar with the content of that report. This report is broken into three main sections: the hardware interfaces, the hardware drivers, and the user software systems.

II. THE HARDWARE INTERFACES

The overall configuration of the connections of the 7912AD transient digitizer to the computer systems is shown in Figure 1. The computer and the Binary Laboratory Interface System (BLIS) are described in Reference 1. Basically, the BLIS units provide 2 sets of totem pole TTL logic signals at the laboratory: 18 input lines, and 18 output lines. These lines appear to the software as individual bits in registers and are programmed as such. In this interfacing, only 15 of the 18 bits are actually used. A transmitter - receiver pair of boards are used since the IEEE-488 interface uses open collector inverted logic. In this scheme, a logical "0" is indicated on a line when the voltage on that line is above 3.5 volts. A logical "1" is indicated by a voltage of less than 0.8 volts. The term "open collector" refers to the internal arrangement of components in the logic gate. Figure 2

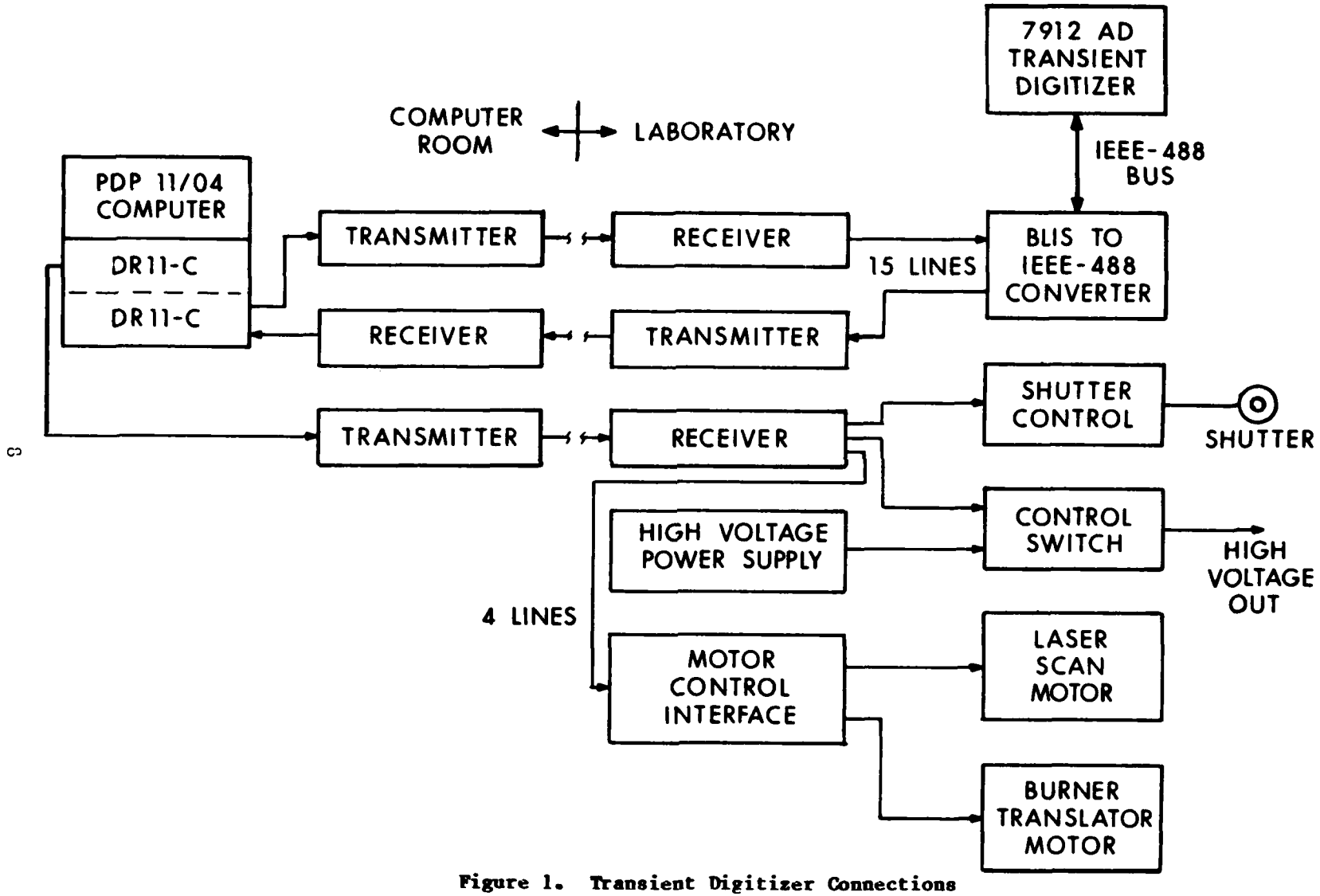


Figure 1. Transient Digitizer Connections

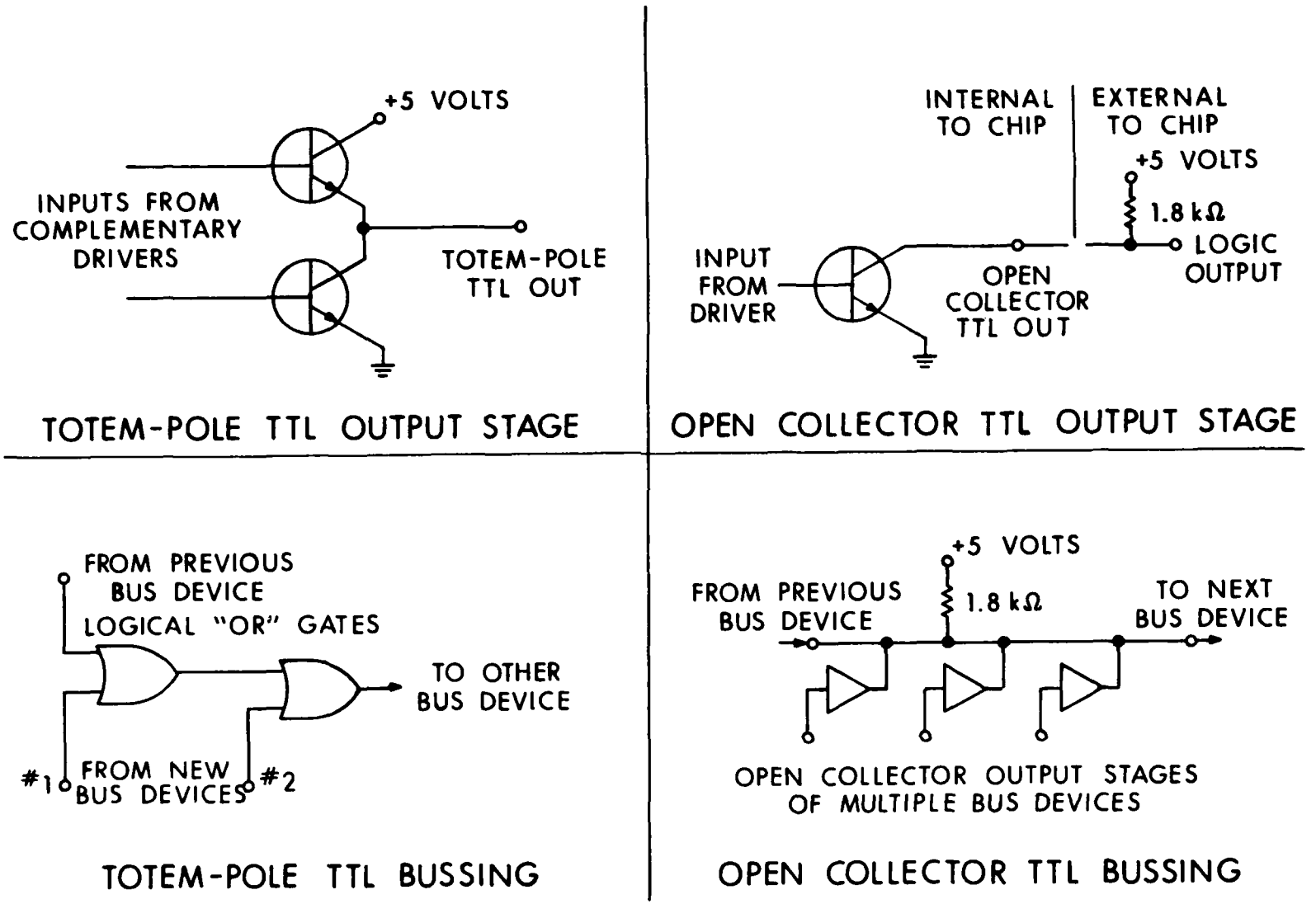


Figure 2. TTL Logic and Bussing

shows the two types of TTL logic and the means of establishing bus structures. In totem-pole TTL, a transistor is used to actively supply current from the circuit's +5 volt supply to the external logic elements on a gate. The low output voltage level is produced by using another transistor to actively clamp the output to the circuit's ground level, sinking any current necessary to do so. In general, a logic gate can source about 0.6 milliamps, and can sink about 16 milliamps. In many cases, it is desirable to be able to allow any one of many gates to assert a logic level on a single line. The logic operation that describes this is the logical "OR". This is particularly useful when dealing with bus structures that are expandable- that is, they may have a variable number of devices connected to the same bus, on the same set of wires in the bus (the IEEE-488 bus is such a structure). In order to do this in totem-pole TTL logic for a bus of 16 lines width, 16 two-input "OR" gates are required for each new device added; one input is for the incoming logic signals from all previous devices, and the other is the logic input from the new device. The output then goes to the next device. This form of bussing is highly undesirable from cost, reliability, complexity, and speed considerations, and is never used in any real systems. The answer to the problem lies in the use of open-collector TTL logic.

The primary difference between totem-pole TTL and open collector TTL is that the pull-up transistor that sources the current in totem pole logic is left out, and the source current is supplied externally to the logic chip by a resistor. This is necessary in order to connect the logic outputs of the gates directly together in a "wired OR" connection. If the output of one totem-pole TTL gate were at a "high" level, and the output of another at a "low" level, and the two wired together, the result would be that the current sourcing transistor would burn out. If, however, the outputs of two open collector gates were connected together, and a pull-up resistor to +5 volts supplied, either could pull the combined output to ground, independent of the other--one OR the other asserts the low condition. This connection, when using inverted logic, is the so-called "wired-OR" connection, and is expandable without adding additional logic. Three, four, five or more gates' outputs may be wired together in such an "OR" connection to form a bus structure. The IEEE-488 bus has 15 different lines, all of which are "wired-OR" connections to the various instruments that are put on the bus, and the bus controller.

As mentioned previously, BLIS is a totem pole TTL logic system. This was done for the purposes of electronic noise immunity. When a TTL line is clamped to ground by a transistor, it is highly immune to electrical noise. Similarly, when clamped to +5 volts by another transistor, it is also highly immune to noise. The IEEE-488 bus is open-collector TTL, and so a box was built that would convert the totem pole signals of BLIS to open collector for the IEEE-488 bus. When a line is clamped to +5 volts through an 1800 ohm resistor, as it is in the IEE-488 bus, noise immunity is greatly reduced and noise induced errors in signals become a problem. For this reason, the IEEE-488 bus must be extremely well shielded, and kept as short as possible. In order to do this, the transient digitizer was either located as close as possible to the BLIS, or the totem pole TTL signals were run to the digitizer, and connected to the converter box, there, so as to minimize the length of IEEE-488 bus. The circuitry of the converter box is shown in Figure 3. Each line from the BLIS receiver board drives a single open collector TTL inverter gate. The output from the gate is looped back to the BLIS driver so that the

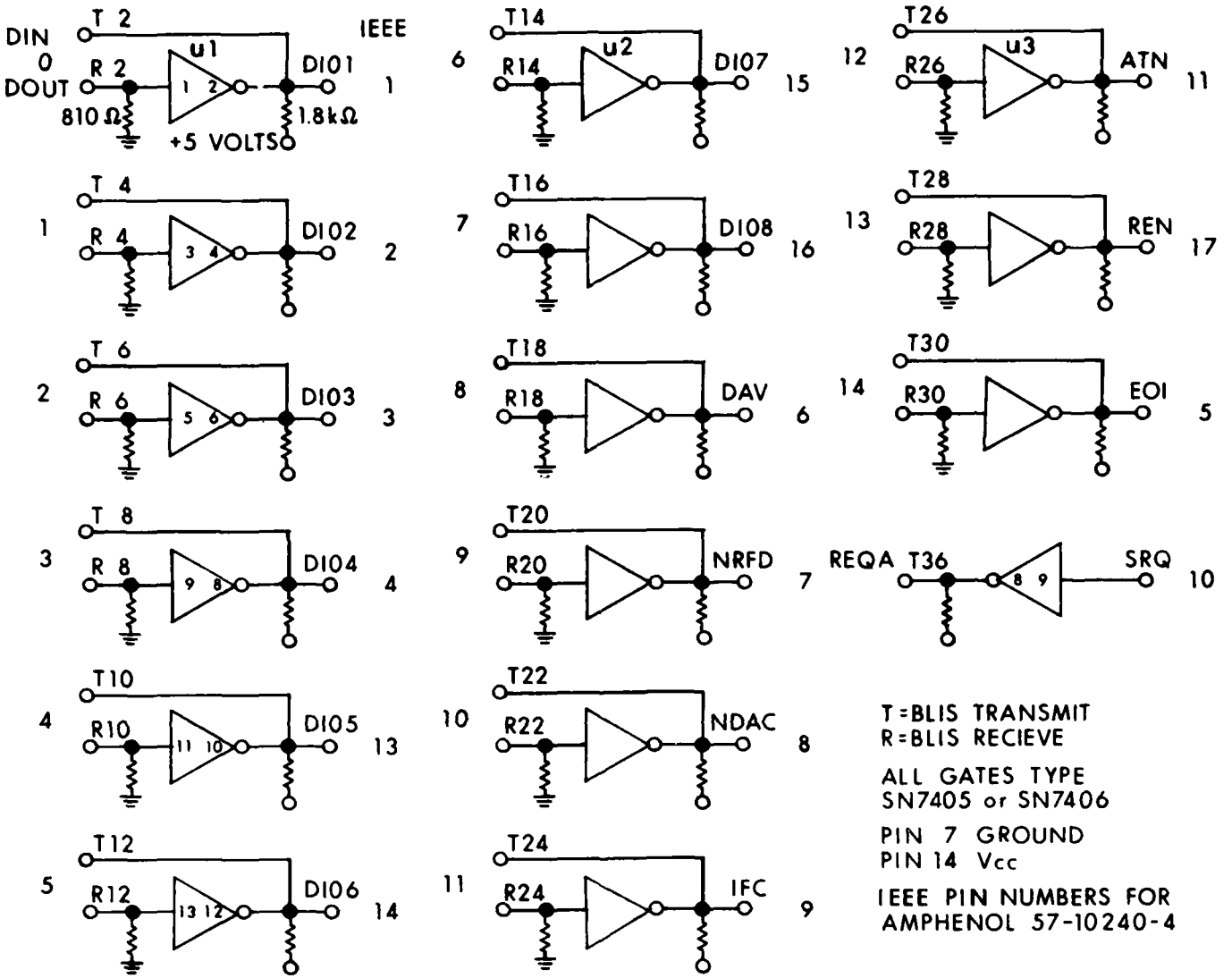


Figure 3. BLIS to IEEE-488 Converter Box

level of the IEEE-488 line may be read. Pull-up resistors are provided in the box as current sources, and to facilitate repair and testing. The computer bus addresses for the input and output registers corresponding to this box are 167754 and 167752 respectively (both addresses given in octal). The bit assignments are as follows:

Output Register : Address 167752 octal

15	14	13	12	10	09	08	07	06	05	04	03	02	01	00

BIT	Signal Name
15	Not Used
14	EOI - End or Identify
13	REN - Remote ENable
12	ATN - Attention
11	IFC - InterFace Clear
10	NDAC - Not Data ACcepted
09	NRFD - Not Ready For Data
08	DAV - Data AVailable
07	DI07 - Data In bit 07
06	DI07 - Data In bit 06
05	DI07 - Data In bit 05
04	DI07 - Data In bit 04
03	DI07 - Data In bit 03
02	DI07 - Data In bit 02
01	DI07 - Data In bit 01
00	DI07 - Data In bit 00

Input Register : Address 167754 octal

15	14	1[illegible]	12	10	09	08	07	06	05	04	03	02	01	00

BIT	Signal Name
15	Not Used
14	EOI* - End or Identify return, inverted
13	REN* - Remote ENable return, inverted
12	ATN* - Attention return, inverted
11	IFC* - InterFace Clear return, inverted
10	NDAC* - Not Data ACcepted return, inverted
09	NRFD* - Not Ready For Data return, inverted
08	DAV* - Data AVailable return, inverted
07	DI07* - Data In bit 07 return, inverted
06	DI07* - Data In bit 06 return, inverted
05	DI07* - Data In bit 05 return, inverted
04	DI07* - Data In bit 04 return, inverted
03	DI07* - Data In bit 03 return, inverted
02	DI07* - Data In bit 02 return, inverted
01	DI07* - Data In bit 01 return, inverted
00	DI07* - Data In bit 00 return, inverted

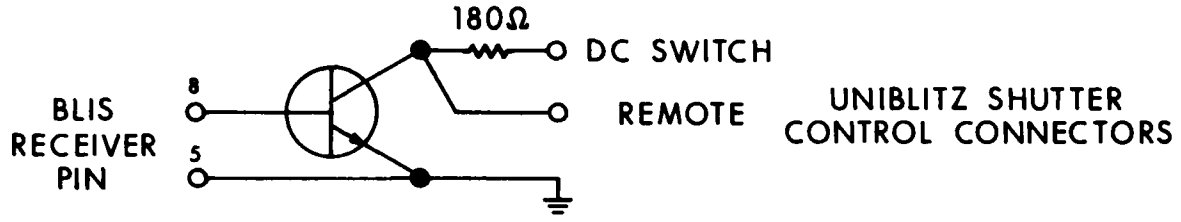
When the IEEE-488 devices are disconnected from the converter box, the turn-around of the output lines to the input lines provides a quick check and convenient means of testing and trouble shooting. The registers can be written and read directly from the computer's console, and if any signals are faulty, they show up in a few seconds. A typical test would be to write all the bits in the output register to zero, and then check the low 15 bits in the return to be all ones. Next, the output register would be written to all ones, and the input low 15 bits checked to be all zeros. Any deviation from this indicates hardware failure. Additional tests on single bits may also be performed. For details on the functions of these signals, the reader is referred to Reference 2.

The additional connections to operate the various other devices in the multiphoton ionization experiment are shown in Figures 4 and 5. The high voltage relay is a coaxial cable relay from Danbury-Knudsen, type CR 72. The coil is for 115 V.A.C., so that a small TTL-drivable relay is used to switch the coil control voltage. This relay is driven directly from the BLIS. The shutter is from D.A. Vincent Associates and goes under the trade name "Uniblitz". The single transistor shown acts as an inverter so that the shutter is normally open when the system is first turned on. The burner position stepper motor and laser scanning stepper motor were both controlled by the same control unit. This control unit is a model MCI-1 from the Quanta-Ray company, which also supplied the laser used for the multiphoton work. The burner was mounted on a translator stage from Velmex, Inc, model B2500, with a Slo-Syn type SS25-1134 stepping motor attached. The stepper motor and cabling in the dye laser are supplied as a normal part of that unit. The wiring of the cables and connection of the motors is shown in Figure 5. The reader is referred to Reference 4 for details concerning the laser control unit not covered here. In order to control all of these additional devices an additional BLIS receiver was used. The bit assignments follow:

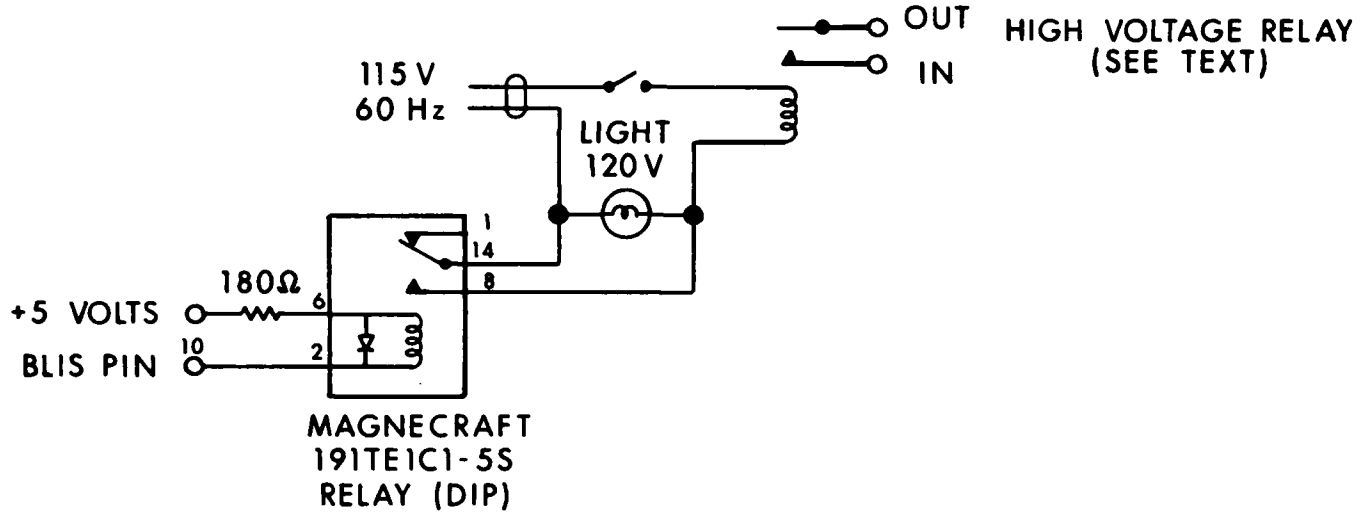
Output Register : Address 167742 octal

15	14	13	12	10	09	08	07	06	05	04	03	02	01	00

BIT	Signal Name
15	Not Used
14	Not Used
13	Not Used
12	Not Used
11	Not Used
10	Not Used
09	Not Used
08	Not Used
07	Not Used
06	Not Used
05	Not Used
04	High Voltage Relay. 0 = turn on H.V., 1 = turn off H.V.
03	Motor Data. 1 = move motor on NDR, 0 = No motor movement
02	Shutter bit. 0 = open shutter, 1 = close shutter
01	Motor address. 1 = Burner motor moves, 0 = Laser scanning motor moves
00	Motor Sign. Clockwise or counterclockwise depending on motor wiring.

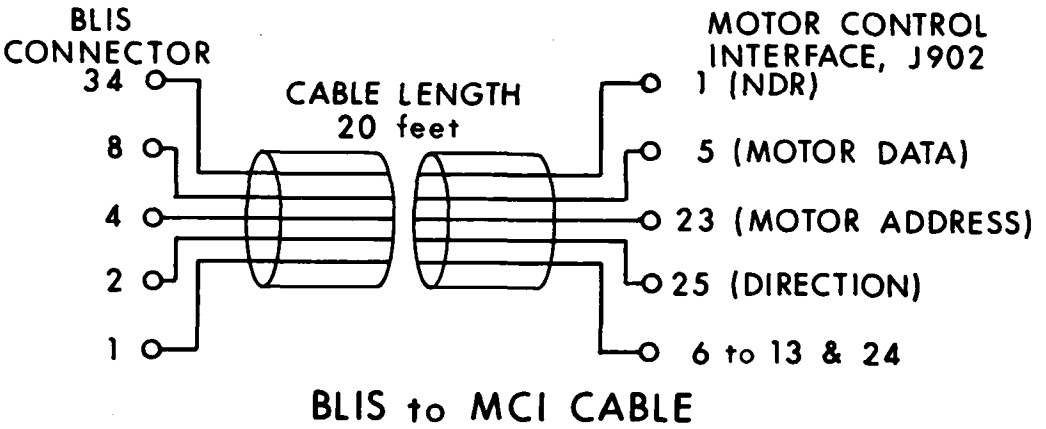


SHUTTER WIRING

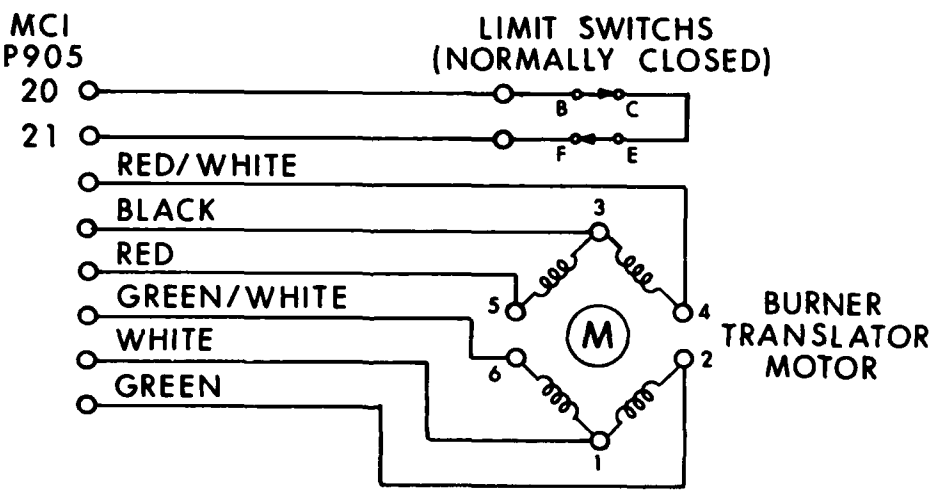


HIGH VOLTAGE SWITCH WIRING

Figure 4. Auxiliary Control Connections



BLIS to MCI CABLE



MCI to BURNER TRANSLATOR CABLE

Figure 5. The Stepper Motor Connections

As can be easily seen, there is much room for additional expansion in this system with 12 bits available for further control functions. At the time of writing, these other bits are unused. The operation of the high voltage relay and shutter are straightforward enough not to require explanation, but such is not the case with the motor stepping. The nature of the BLIS is that whenever the output register is loaded by the computer, a special signal line called "New Data Ready" (NDR) has a pulse placed upon it. This is used by the motor controller to initiate the stepping of a motor. In order to inhibit the stepping of a motor when either the shutter or high voltage relay is operated, a motor data line is provided. If the motor data is zero, then zero steps are initiated in any motor. If the motor data is 1, then one step will be initiated when the NDR signal is received from BLIS by the controller. The motor direction and address are then self explanatory. The pinouts of the cables are given in the Figures.

III. THE HARDWARE DRIVER ROUTINES

The hardware driver routines for the system were written in MACRO-11, the available assembly language on the PDP-11 computer systems in use, under the RT-11 operating system. They were written to support the standard calling protocol for the FORTRAN compiler on that same system, so that all are FORTRAN callable as subroutines. The simplest routines to present are those to operate the shutter, high voltage relay, laser scanner, and burner movement stepper motor. These are named "SHUTTR", "JOLTS", "LASER", and "MOVBUR" respectively. It is possible to connect any other devices that can be controlled by TTL voltage levels to these lines and use the same software to drive them. The source code follows.

```
;*************************************************************************
;* SHUTTR.MAC THE SHUTTER CONTROLLER FOR MULTIP.MIZ. CALLED BY:          *
;*                                                                       *
;*       CALL SHUTTR(IARG)                                               *
;*                                                                       *
;*              IARG = 0  TO OPEN SHUTTER                                *
;*              IARG = 1  TO CLOSE SHUTTER                               *
;*************************************************************************

        .TITLE  SHUTTR
        .GLOBL  SHUTTR
        .MCALL  .REGDEF
        .REGDEF
SHUTTR: BIC     #14,@#167762    ;OPEN SHUTTER,TURN OFF MOTOR DATA BIT
        TST     (R5)+           ;INC R5 BY 2
        TST     @(R5)+          ;GET ARGUMENT IARG
        BEQ     DONE            ;GO TO DONE IF ZERO
        BIS     #4,@#167762     ;CLOSE THE SHUTTER
DONE:   RTS     PC
        .END


;*************************************************************************
;* JOLTS.MAC THE HIGH VOLTAGE ION COLLECTOR CONTROLLER FOR MULTIP.MIZ.    *
;* CALLED BY:                                                            *
;*                                                                       *
;*       CALL JOLTS(IARG)                                                *
;*                                                                       *
;*              IARG = 0  TO TURN ON ION COLLECTOR H.V                   *
;*              IARG = 1  TO TURN OFF ION COLLECTOR H.V                  *
;*************************************************************************

        .TITLE  JOLTS
        .GLOBL  JOLTS
        .MCALL  .REGDEF
        .REGDEF
JOLTS:  BIC     #30,@#167762    ;TURN ON H.V. ON ION COLLECTOR
        TST     (R5)+           ;INC R5 BY 2
        TST     @(R5)+          ;GET ARGUMENT IARG
        BEQ     DONE            ;GO TO DONE IF ZERO
        BIS     #20,@#167762    ;TURN ON H.V. TO ION COLLECTOR
DONE:   MOV     #77777,R0       ;SET UP DELAY LOOP TO ALLOW RELAY TO SETTLE
LOOP:   DEC     R0
        BNE     LOOP
        RTS     PC
        .END
```

```
;*************************************************************************
;* LASER.MAC    THE ROUTINE FOR SCANNING THE PULSED DYE LASER OF THE      *
;*              MULTIPHOTON IONIZATION EXPERIMENT. THE FORM OF THE CALL   *
;*              IS:                                                       *
;*                      CALL LASER(NSTEPS)                                *
;*              WHERE NSTEPS IS A POSITIVE OR NEGATIVE NUMBER OF .012     *
;*              NANOMETER WAVELENGTH STEPS.                               *
;*************************************************************************

        .TITLE  LASER
        .GLOBL  LASER
        .MCALL  .REGDEF
        .REGDEF
        .ENABL  LSB
LASER:  BIC     #13,@#167762    ;CLR SIGN,MOTOR ADDRESS,AND MOTOR DATA BITS
        TST     (R5)+           ;INC R5 BY 2
        MOV     @(R5)+,R0       ;GET NUMBER OF STEPS, NSTEPS
        NEG     R0              ;NEGATE FOR PROPER DIRECTION
        BPL     3$              ;GO TO 3$ IF POSITIVE NUMBER
        NEG     R0              ;MAKE IT POSITIVE
        BIS     #1,@#167762     ;SET SIGN(DIRECTION) BIT
3$:     BIS     #10,@#167762    ;STEP THE MOTOR ON THE LASER
        MOV     #77777,R1       ;SET UP DELAY SO WE DON'T STEP TOO FAST
2$:     DEC     R1
        BNE     2$
        DEC     R0              ;COUNT DOWN NUMBER OF STEPS
        BGE     3$              ;AND DO MORE STEPS IF NOT ZERO
        BIC     #13,@#167772    ;TURN OFF THE LINES
        RTS     PC
        .END


;*************************************************************************
;* MOVBUR.MAC   THE ROUTINE THAT MOVES THE STEPPER MOTOR CONNECTED TO THE *
;*              TRANSLATION STAGE THAT THE BURNER SYSTEM IS MOUNTED UPON. *
;*              THE FORM OF THE CALL IS:                                  *
;*                                                                        *
;*                              CALL MOVBUR(ISTEP)                        *
;*                                                                        *
;*              WHERE ISTEP IS A POSITIVE OR NEGATIVE NUMBER OF STEPS OF  *
;*              THE MOTOR. ONE STEP = 0.00025 INCHES MOVEMENT             *
;*************************************************************************

        .TITLE  MOVBUR
        .GLOBL  MOVBUR
        .MCALL  .REGDEF
        .REGDEF
        .ENABL  LSB
MOVBUR: BIC     #13,@#167762    ;CLR SIGN,MOTOR ADDR, DATA  BITS
        TST     (R5)+           ;INC R5 BY 2
        MOV     @(R5)+,R0       ;GET NUMBER OF STEPS
        BPL     1$              ;GO TO 1$ IF POSITIVE NUMBER
        NEG     R0              ;MAKE IT POSITIVE
```

```
        BIS     #1,@#167762     ;SET SIGN(DIRECTION) BIT
1$:     ASL     R0              ;MULTIPLY BY TWO(SINCE 200 STEPS=400 PULSES)
3$:     BIS     #12,@#167762    ;STEP THE MOTOR ON THE BURNER
        MOV     #3000,R1        ;SET UP DELAY
2$:     DEC     R1
        BNE     2$
        DEC     R0              ;COUNT DOWN NUMBER OF STEPS
        BGE     3$              ;AND DO MORE STEPS IF NOT ZERO
        BIC     #13,@#167772    ;TURN OFF THE LINES
        RTS     PC
        .END
```

At this point it is appropriate to explain the decision of not using the commercially available IEEE-488 to PDP-11 computer interface, instead of the custom one used in this work. It is our philosophy to keep computers (being rather electrically sensitive pieces of equipment) away from hostile and electrically noisy environments. The pulsed lasers in use are notorious generators of intense wide spectrum electronic noise. In addition, the burner systems are sources of corrosive gasses. In order to use the commercial interface, the computer must be within a few meters of the experiment in order not to exceed the transmission range of IEEE-488 within this noisy environment. By going to our BLIS units for the long distance driving of signals, the computers could be placed in a safe environment and higher noise immunity gained. In addition, since the central site contains four equivalent machines and patching systems, it is possible for one machine to break down and another take over with a trivial change of patch cords.

In this paragraph we begin discussion of the IEEE-488 control system and transient digitizer command package. These two software packages are not treated as a single unit because there is a general portion, and a device specific portion. The general portion consists of a routine to provide the necessary manipulation of control signals in order to transmit information to devices on the IEEE-488 bus, another to provide the manipulation of control signals for receiving information from the bus, and a third to provide a programming language for bus operations. The names of these three routines are "TALKIT", "LSNR", and "CIF" respectively. The device specific portion of the package consists of a series of so-called "Operation Definition Blocks" which define the character strings that represent commands to a given instrument, and "Procedure Definition Blocks" which are lists of ODB's that, when sent in the order given, make the instrument perform complete tasks. These PDB's are sent to the CIF routine as arguments of the subroutine CIF call. Figure 6 illustrates the overall architecture of the system. The reason for building the system in this manner is to provide as general as possible a method for interfacing any IEEE-488 instrument to the system, not simply a 7912AD transient digitizer. The only things that need be changed to implement another instrument are the ODB's and PDB's, both of which have the simplest of formats, as will be shown.

In order to begin the description of the LSNR routine and TALKIT routine, it is well to review the IEEE-488 protocol for communication. To begin, the sender (computer) asserts REN (low), asserts ATN, and places the listening device's listening address on the data lines. This listening address is unique to each instrument on the IEEE bus, and is hardware set prior to connection to the bus. When the device sees its address in this way, it comes

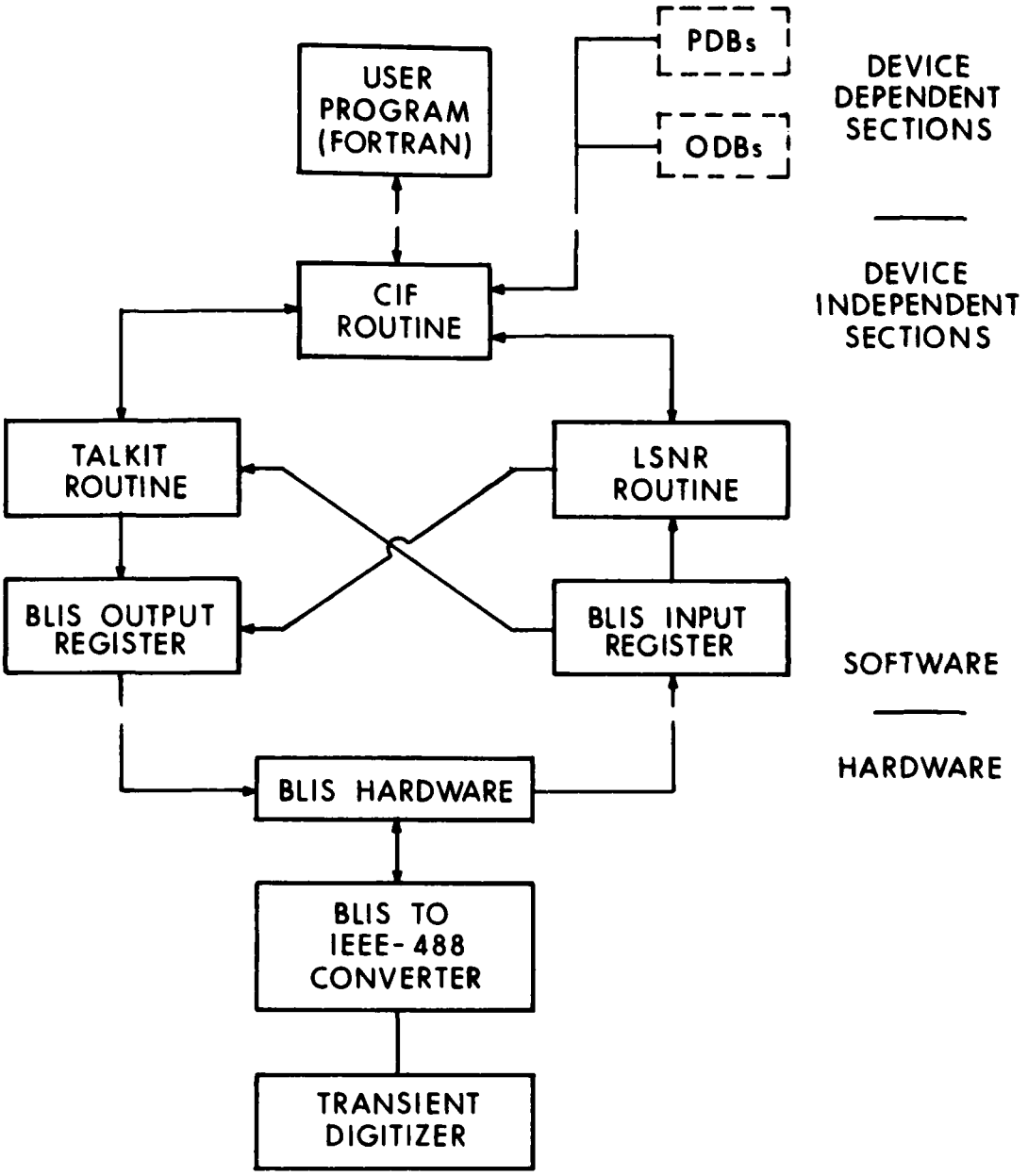


Figure 6. The Architecture of the IEEE-488 Control System

on line as a listener by asserting the two lines NRFD and NDAC. When the sender sees this, it releases the ATN line and the data lines but leaves REN asserted. Releasing REN at any time causes all devices on the bus to go off line. The second step of the handshake begins by the sender placing the data byte on the data lines. At this point, the listening device indicates that it is ready to accept the data releasing the NRFD line. The sender then continues the handshake by asserting DAV, which indicates to the listener that the data is valid and is to be accepted. The listener then asserts NRFD while it accepts the data byte, after which it releases NDAC to indicate it is done accepting the data. The sender then releases DAV, which causes the listener to assert NDAC, and the initial condition is re-established for the next transmission. When the LSNR routine is active, the computer takes on the task of the listener, and the remote instrument becomes the sender ("talker" in the IEEE-488 jargon). The protocol is unchanged in this mode. The actual source codes, again in MACRO-11, for the listener and talker routines follow:

```
        .TITLE  LSNR
        .GLOBL  LSNR,DWA,ENDDEL,BYTCNT,EOIFLG,CKSUMF    ; see routine
                                                        ;"CIF" for definitions
        .MCALL  .REGDEF,.PRINT
        .REGDEF
        .ENABL  AMA
        .ENABL  LSB
         EOI = 40000
         NDAC = 2000
         NRFD = 1000
         DAV = 400
         DRIN = 167754
         DROUT = 167752
;
LSNR:   CLR     NOSTOR
        CMP     #177777,DWA     ;SEE IF NOT GOING TO STORE DATA
        BNE     1$              ;GO TO 1$ IF GOING TO STORE
        MOV     #1,NOSTOR       ;SET NO STORE FLAG
1$:     MOV     DWA,R1          ;GET LOCAL COPY OF ADDRESS FOR DATA
        INC     R1              ;SINCE 7912 SENDS HIGH BYTE FIRST
        CMP     #177777,ENDDEL  ;SEE IF INPUT DELIMITER IS FLAG
        BNE     INPDEL          ;AND INPUT UNTIL DELIMITER IS REACHED
        TST     BYTCNT          ;SEE IF NO BYTES TO GET
        BEQ     ERROR           ;ERROR IF SO..
        MOV     BYTCNT,R2       ;GET LOCAL COPY OF BYTE COUNT
2$:     JSR     PC,INAWRD       ;GET AN INPUT WORD
        TST     EOIFLG          ;WE GOING TO LOOK FOR UNEXPECTED EOI?
        BNE     7$              ;NO. GO TO 7$
        BIT     #EOI,INWORD     ;SEE IF EOI IS SET
        BEQ     ENDERR          ;IF SO, UNEXPECTED END OF MESSAGE ERROR
7$:     TST     NOSTOR          ;WE GOING TO STORE IT?
        BNE     3$              ;GO TO 3$ IF NOT
        JSR     PC,STOR         ;STORE IT OTHERWISE
3$:     DEC     R2              ;COUNT DOWN BYTECOUNT
        BNE     2$              ;AND CONTINUE MORE INPUT 'TIL DONE
        BR      DONE            ;RETURN IF DONE
INPDEL: JSR     PC,INAWRD       ;GET AN INPUT WORD
        CMPB    ENDDEL,INWORD   ;SEE IF WE'VE REACHED THE END DELIMITER
        BNE     4$              ;GO TO 4$ IF NOT
```

```
        BR      DONE            ;ELSE, RETURN
4$:     TST     EOIFLG          ;GOING TO IGNORE EOI?
        BNE     8$              ;YES? THEN GO TO 8$
        BIT     #EOI,INWORD     ;CHECK FOR BAD EOI
        BEQ     ENDERR          ;DO ERROR THING IF SO
8$:     TST     NOSTOR          ;SEE IF GOING TO STORE DATA
        BNE     INPDEL          ;GET NEXT ONE IF NOT
        JSR     PC,STOR         ;ELSE, STORE IT
        BR      INPDEL          ;AND GO GET THE NEXT ONE
;********* SUBROUTINE STOR SINCE HIGH BYTE IS SENT FIRST BY 7912 **********
STOR:   MOVB    INWORD,@R1      ;SOCK AWAY THE BYTE
        MOVB    INWORD,R3       ;GET THE DATA BYTE INTO R3
        BIC     #100000,R3      ;GET RID OF ANY SIGN EXTENDING
        ADD     R3,CKSUMF       ;AND ADD IT TO THE CHECKSUM FOUND
        CMP     R1,DWA          ;SEE IF LOWBYTE IS STORED YET
        BEQ     5$              ;GO TO 5$ IF YES
        DEC     R1              ;POINT AT LOW BYTE STORE LOCATION IF NOT
        BR      6$              ;GO TO 6$
5$:     ADD     #2,DWA          ;POINT DWA AT NEXT WORD LOCATION
        MOV     DWA,R1          ;GET COPY IN R1
        INC     R1              ;POINT R1 AT HIGH BYTE
6$:     RTS     PC              ;AND RETURN
;
;  ERROR STUFF.....
ERROR:  .PRINT  #BADRED         ;ERROR ON BYTECOUNT
         BR      DONE
BADRED: .ASCIZ  /ERROR..BYTECOUNT=0 ON READ/
        .EVEN
ENDERR: .PRINT  #BADERR
         BR      DONE
BADERR: .ASCIZ  /ERROR..UNEXPECTED EOI ON READ/
        .EVEN
INWORD: .WORD   0
NOSTOR: .WORD   0
DONE:    MOV     #20000,DROUT    ;LEAVE REN SET, AND THEN RETURN
         JSR     PC,DELY         ;ALLOW TO SETTLE
         CLR     NOSTOR
         RTS     PC
        .DSABL  LSB
;
;  SUBROUTINE INAWRD  ....THIS IS THE LISTENER HANDSHAKE ROUTINE
;
        .ENABL  LSB
INAWRD: BIS     #NRFD,DROUT     ;SET NRFD
        BIS     #NDAC,DROUT     ;ASSERT NDAC
        BIC     #NRFD,DROUT     ;CLEAR NRFD
        JSR     PC,DELY         ;ALLOW TO SETTLE
1$:     BIT     #DAV,DRIN       ;CHECK DAV
        BNE     1$              ;LOOP UNTIL ASSERTED
        BIS     #NRFD,DROUT     ;SET NRFD
        JSR     PC,DELY         ;ALLOW TO SETTLE
        MOV     DRIN,INWORD     ;GET THE INPUT WORD
        COMB    INWORD          ;IEEE STUFF IS SENT INVERTED, SO COMPLEMENT.
        BIC     #NDAC,DROUT     ;CLEAR NDAC
```

```
2$:     BIT     #DAV,DRIN       ;LOOK AT DAV
        BEQ     2$              ;LOOP UNTIL NOT ASSERTED(HIGH)
        BIS     #NDAC,DROUT     ;ASSERT NDAC
        JSR     PC,DELY         ;ALLOW TO SETTLE
        RTS     PC              ;RETURN
DELY:   MOV     R1,S1
        MOV     #10,R1
LP:     DEC     R1
        BNE     LP
        MOV     S1,R1
        RTS     PC              ;SUPER-SHORT DELAY FOR BLIS SETTLING
S1:     .WORD   0
        .END
```

The code for the talking routine to send data to the IEEE device from the computer is:

```
        .TITLE  TALKIT
        .GLOBL  TALKIT,OUTWRD,WAITFL
        .MCALL  .REGDEF,.PRINT
        .REGDEF
        .ENABL  LSB
        .ENABL  AMA
;
        DROUT = 167752
        DRIN  = 167754
        REN   = 20000           These are the bits corresponding to the lines
        ATN   = 10000
        NDAC  = 2000
        NRFD  = 1000
        DAV   = 400
        EOI   = 40000
;
TALKIT: MOV     R0,-(SP)        ;PUSH R0
        MOV     @#OUTWRD,DROUT  ;OUTPUT THE WORD
        JSR     PC,DELY         ;ALLOW TO SETTLE
        MOV     #177777,R0      ;LOAD COUNTER
1$:     BIT     #NRFD,DRIN      ;CHECK FOR NRFD
        BNE     2$              ;GO TO 2$ IF NRFD HIGH
        TST     WAITFL          ;SEE IF NOT GOING TO TIME OUT
        BNE     1$              ;AND LOOP IF SO
        DEC     R0              ;DECREMENT COUNTER
        BEQ     ERR1$           ;NEVER DID GO HIGH TIME OUT
        BR      1$              ;GO BACK AND WAIT SOME MORE
2$:     MOV     #177777,R0      ;LOAD COUNTER AGAIN
3$:     BIT     #NDAC,DRIN      ;CHECK NDAC
        BEQ     4$              ;GO TO 4$ IF LOW
        DEC     R0              ;COUNT DOWN
        BEQ     ERR3$           ;TIMEOUT WAITING FOR ACRS..ERROR!
        BR      3$              ;WAIT SOME MORE UNTIL TIMEOUT
4$:     BIS     #DAV,DROUT      ;SET DAV LINE
        JSR     PC,DELY         ;ALLOW TO SETTLE
6$:     MOV     #177777,R0      ;LOAD COUNTER
9$:     BIT     #NDAC,DRIN      ;CHECK NDAC
```

```
        BNE     5$              ;CONTINUE TO 5
        DEC     R0              ;COUNT DOWN COUNTER
        BNE     9$              ;LOOP UNTIL HIGH OR TIMEOUT
        BR      ERROUT          ;TIMEOUT ERROR
5$:     BIC     #DAV,DROUT      ;TURN OFF DAV
        JSR     PC,DELY         ;ALLOW TO SETTLE
10$:    BIC     #ATN,DROUT      ;AND ATN IF ASSERTED
        JSR     PC,DELY         ;ALLOW TO SETTLE
        BIC     #EOI,DROUT      ;AND ALSO EOI.
        JSR     PC,DELY         ;ALLOW TO SETTLE
        CLRB    DROUT           ;DON'T LEAVE GARBAGE IN DROUT LOW BYTE EITHER
        JSR     PC,DELY         ;ALLOW TO SETTLE
        BR      8$              ;AND LEAVE....
ERROUT: .PRINT  #ERRMSG         ;HERE'S THAT ERROR THING
8$:     MOV     (SP)+,R0        ;RESTORE R0!
        RTS     PC              ;AND RETURN TO CALLER
ERRMSG: .ASCIZ  /TALKIT ERROR...NO LISTENER/
        .EVEN
ERR1$:  .PRINT  #ER1
         BR      8$
ER1:    .ASCIZ  /ERROR TALKIT 1$/
        .EVEN
ERR3$:  .PRINT  #ER3$
         BR      8$
ER3$:   .ASCIZ  /ERROR TALKIT 3$/
        .EVEN
DELY:    RTS     PC             ;SUPER-SHORT SETTLING DELAY
        .END
```

Both of the above routines form the lowest level of the software between the user and the hardware instrumentation. Essentially, they allow the relatively simple hardware interface to appear as a more sophisticated one to the next higher level of software, the "CIF" routine. The function of the CIF (for Control InterFace) routine is to provide an "assembly" language to drive and manipulate the hardware in such a way as to require minimum memory to store the routines, and provide device independence, so as to be general for any IEEE instrument. As IEEE-488 instruments accept commands in the form of alphanumeric characters, this routine parses strings of these characters and checks for embedded control sequences to perform special functions. All other characters are sent as ASCII strings to the devices on the bus as commands. The format of these strings is defined by the PDBs and the ODBs. An ODB consists of a globally named starting point that contains the first character in that operation string, followed by the remaining characters in the string, and terminated by a null byte. A PDB consists of a list of those global ODB names, terminated by a null word. The PDB is also given a global name, as was the ODB. The reason for this structure is that if 4 or 5 PDBs all refer to a single ODB, that ODB need appear in memory only once under this scheme. If, instead, all procedures were defined as complete strings, multiple copies of a given sub-string would appear in memory and waste valuable memory space.

In order to get a single character from a procedure, first the PDB is referenced to get the first ODB name. Next, the ODB is read byte by byte until a null byte is detected. Upon detecting a null byte, the next ODB name is read from the PDB. If this name is not the null word signifying the end of

the PDB, the next ODB is opened, and the reading continues. If the end of the PDB is found, a return to the calling routine from the CIF routine is initiated. When a character is read in from an ODB, it first is checked to see if it is a special control character (see the program listing for the meanings and definitions of the control characters). If it is a control character, appropriate action is taken. If it is not a control character, it is sent to the "TALKIT" routine to be output to the devices on the IEEE bus. As a special case, the control character " < " causes the computer to become a listener by invoking the "LSNR" routine. The source code for the CIF routine follows:

```
;*********************************************************************
;* CIF.MAC    THIS IS THE CONTROL INTERFACE SUBROUTINE FOR THE TEKTRONIX *
;*              7912AD TRANSIENT DIGITIZER CONTROL PACKAGE. THIS FORTRAN *
;*              CALLABLE ROUTINE ACCEPTS A LIST OF NAMES(I.E. ADDRESSES) *
;*              OF PROCEDURE DEFINITION BLOCKS (PDB'S) THAT CONSIST OF   *
;*              NAMES(ADDRESSES) OF OPERATION DEFINITION BLOCKS(ODB'S).  *
;*              THE CIF ROUTINE THEN READS, BYTE BY BYTE, THE ODB, AND   *
;*              CHECKS FOR CONTROL CHARACTERS. THE ROUTINE THEN EITHER   *
;*              SETS OR CLEARS CONTROL BITS IN AN OUTPUT WORD, OR CALLS  *
;*              THE TALKER ROUTINE TO OUTPUT THE ASSEMBLED OUTPUT WORD.  *
;*              THE CONTROL CHARACTERS ARE:                              *
;*                                                                       *
;*              <       BECOME LISTENER, GET INPUT STRING FROM 7912AD    *
;*              ~       TURN OFF LINE REPRESENTED BY NEXT CHARACTER      *
;*              _       INTERPRET NEXT CHARACTER AS AN OPERATION CODE    *
;*              >       SET LISTENER ACTIVE FLAG                         *
;*              !       TURN ON OR OFF REN LINE (OFF IF PRECEDED BY ~)   *
;*              $       TURN ON OR OFF ATN LINE (OFF IF PRECEDED BY ~    *
;*              *       DO AN INTERFACE CLEAR                            *
;*              .       TURN ON EOI LINE (OFF IF PRECEDED BY ~)          *
;*                                                                       *
;*                 OPERATION CODES, IF PRECEDED BY _                     *
;*              B       LOAD BYTECOUNT WORD WITH NEXT ONE BYTE           *
;*              D       LOAD DATA ARRAY ADDRESS WORD WITH NEXT 2 BYTES   *
;*              E       LOAD END DELIMITER WORD WITH NEXT 2 BYTES        *
;*              C       DO CHECKSUM TEST, PRINT ERROR MESSAGE IF FAIL    *
;*              I       ENABLE/DISABLE INTERRUPT ON SRQ (~_I=DISABLE)    *
;*              R       REDUCE BYTECOUNT WORD BY 1                       *
;*              F       FLAG(IGNORE) EOI CONCURRENT WITH INPUT           *
;*              W       WAIT FOR NRFD TO CLEAR, DO NOT TIME OUT ERROR    *
;*********************************************************************
;
        .TITLE  CIF
        .GLOBL  CIF,TALKIT,LSNR,OUTWRD,BYTCNT,ENDDEL,DWA,CKSUMR,CKSUMF,EOIFLG
        .GLOBL  WAITFL
        .MCALL  .REGDEF,.PRINT
        .REGDEF
        CSR = 167750
        DROUT = 167752
        .ENABL  LSB             ;ENABLE THE USE OF LOCAL SYMBOLS
;
CIF:    MOV     (R5)+,NARG      ;FETCH THE NUMBER OF PDB'S TO PROCESS
        JSR     PC,FPPDB        ;FETCH THE POINTER TO THE PDB
```

```
        JSR     PC,FPODB        ;FETCH THE POINTER TO THE ODB
BEGIN:  JSR     PC,GETBYT       ;GET A BYTE FROM ODB,STUFF IN R0
        TST     DONFLG          ;DONE?
        BEQ     1$              ;NO? THEN CONTINUE
        RTS     PC              ;DONE? THEN RETURN.
1$:     CMPB    #74,R0          ;IS THE BYTE A "<" ?
        BNE     2$              ;IF NOT, GO TO 2$
        JSR     PC,LSNR         ;IF SO, THEN CALL THE LISTEN SUBROUTINE
        BR      BEGIN           ;AND THEN BEGIN AGAIN
2$:     CMPB    #176,R0         ;IS IT A "~" ?
        BNE     3$              ;NO? GO TO 3$
        MOV     #1, OFFLAG      ;SET THE "OFF FLAG"
        BR      BEGIN           ;AND RESTART
3$:     CMPB    #134,R0         ;IS IT A "_" ?
        BNE     4$              ;NO? GO TO 4$
        MOV     #1,OPFLAG       ;SET THE OPERATION FLAG
        BR      BEGIN           ;GO TO BEGIN
4$:     CMPB    #76,R0          ;IS IT A ">" ?
        BNE     5$              ;NO? GO TO 5$
        MOV     #1,@#LAF        ;SET THE LISTENER ACTIVE FLAG
        TST     OFFLAG          ;IS OFF FLAG SET?
        BEQ     BEGIN           ;GO TO BEGIN IF NOT
        CLR     OFFLAG          ;TURN OFF OFF FLAG IF SET
        CLR     LAF             ;TURN OFF LISTENER ACTIVE FLAG
        JMP     BEGIN           ; GO TO BEGIN
5$:     CMPB    #41,R0          ;IS IT A "!" ?
        BNE     6$              ;NO? GO TO 6$
        TST     OFFLAG          ;IS OFF FLAG SET?
        BEQ     7$              ;NO? GO TO 7$
        BIC     #20000,OUTWRD   ;CLEAR THE REN LINE IN THE OUT WORD
        JMP     BEGIN           ;AND GO TO BEGIN
7$:     BIS     #20000,OUTWRD   ;SET THE REN LINE IN THE OUT WORD
        JMP     BEGIN           ;AND GO TO BEGIN
6$:     CMPB    #44,R0          ;IS IT A "$" ?
        BNE     8$              ;NO? GO TO 8$
        TST     OFFLAG          ;IS OFF FLAG SET?
        BNE     9$              ;YES? GO TO 9$
        BIS     #10000,OUTWRD   ;NO? THEN SET THE ATN BIT IN OUTWRD
        JMP     BEGIN           ;GO TO BEGIN
9$:     BIC     #10000,OUTWRD   ;COME HERE IF OFF FLAG SET AND CLEAR ATN
        CLR     OFFLAG          ;TURN OFF THE OFF FLAG
        JMP     BEGIN           ;GO TO BEGIN
8$:     CMPB    #56,R0          ;IS IT A "." ?
        BNE     11$             ;NO? GO TO 11$
        TST     OFFLAG          ;IS OFF FLAG SET?
        BNE     10$             ;YES? GO TO 10$
        BIS     #40000,OUTWRD   ;SET EOI LINE IN OUTWORD
        JMP     BEGIN           ;GO TO BEGIN
10$:    BIC     #40000,OUTWRD   ;CLEAR EOI IN OUTWORD
        CLR     OFFLAG          ;CLEAR THE OFF FLAG
        JMP     BEGIN           ;GO TO BEGIN
11$:    CMPB    #52,R0          ;IS IT A "*" ?
        BNE     12$             ;NO? GO TO 12$
        MOV     #1000,R1        ;SET UP TIMING CONSTANT FOR LOOP
```

```
        BIS     #4000,@#DROUT   ;TURN ON IFC LINE IN DROUT DIRECTLY
13$:    DEC     R1              ;BEGIN TIMING LOOP
        BNE     13$             ;AND LOOP TIL R1=0
        BIC     #4000,DROUT     ;TURN OFF IFC LINE IN DROUT
        JMP     BEGIN           ;GO TO BEGIN
12$:    TST     OPFLAG          ;IS THE OPERATION FLAG SET?
        BNE     14$             ;NO? THEN GO TO 14$
        BIT     #20000,OUTWRD   ;IS REN SET?
        BEQ     ERR1            ;IF NOT, THEN PRINT THE ERROR STUFF
        TST     LAF             ;IS THERE AN ACTIVE LISTENER ?
        BEQ     ERR1            ;ERROR IF NO ACTIVE LISTENER
        MOVB    R0,OUTWRD       ;PUT THE BYTE IN OUT WORD FOR OUTPUT
        JSR     PC,TALKIT       ;TALK THE OUT WORD
        JMP     BEGIN           ;GO TO BEGIN
ERR1:   .PRINT  #ERRMSG         ;TYPE THE ERROR MESSAGE
        RTS     PC              ;RETURN TO CALLER
ERRMSG: .ASCIZ /ERROR! NO LISTENERS!!/
        .EVEN
14$:    CMPB    #102,R0         ;IS IT A "B" ?
        BNE     15$             ;NO? GO TO 15$
        MOV     #BYTCNT,R1      ;LOAD POINTER WITH ADDRESS OF BYTECOUNT
        JSR     PC,GETBYT       ;GET ONE BYTE
        MOVB    R0,(R1)+        ;PUT IT IN LOWBYTE OF BYTECOUNT
        CLRB    (R1)+           ;AND CLEAR THE UPPER BYTE
        BR      21$             ;CLEAN UP AND RETURN
15$:    CMPB    #104,R0         ;IS IT A "D" ?
        BNE     16$             ;NO? GO TO 16$
        MOV     #DWA,R1         ;LOAD POINTER WITH DATA WORD ADDRESS WORD AD.
        BR      17$             ;GO TO 17$
16$:    CMPB    #105,R0         ;IS IT AN "E"
        BNE     18$             ;NO? GO TO 18$
        MOV     #ENDDEL,R1      ;ADDRESS OF END DELIMITER TO R1
17$:    JSR     PC,GETBYT       ;GET A BYTE FROM CURRENT ODB
        MOVB    R0,(R1)+        ;STORE THE LOW BYTE IN THE APPROPRIATE LOCA-
                                ; TION FOR THE DWA,ENDDEL,OR BYTCNT
        JSR     PC,GETBYT       ;DO THE PREVIOUS 2 STEPS AGAIN
        MOVB    R0,(R1)+        ;FOR THE HIGH BYTE
        BR      21$             ;DONE WITH OPERATION,GO TO 21$
18$:    CMPB    #103,R0         ;IS IT A "C" ?
        BNE     19$             ;NO? GO TO 19$
        ADD     CKSUMR,CKSUMF   ;ADD THE RECEIVED CHECKSUM TO THE SENT ONE
        TSTB    CKSUMF          ;LOWBYTE = 0 ?
        BEQ     21$             ;GO TO 21$ IF O.K.
        .PRINT  #ERR2           ;ELSE PRINT ERROR MESSAGE
21$:    CLR     OPFLAG          ;DONE WITH THIS OPERATION
        JMP     BEGIN           ;GO TO BEGIN
ERR2:   .ASCIZ  /ERROR!! CHECKSUM ERROR!/
        .EVEN
19$:    CMPB    #111,R0         ;IS IT AN "I" ?
        BNE     20$             ;NO ? GO TO 20$
        TST     OFFLAG          ;IS OFF FLAG SET?
        BNE     22$             ;YES? GO TO 22$
        BIS     #100,CSR        ;TURN ON THE INTERRUPT ENABLE IN THE DR11C
        BR      21$             ;DONE, SO CLEAN UP AND GO..
```

```
22$:    BIC     #100,CSR        ;CLEAR THE I.E. BIT IN THE DR11C
        CLR     OFFLAG          ;CLEAR THE OFF FLAG
        BR      21$             ;DONE, SO CLEAN UP AND GO..
20$:    CMPB    #122,R0         ;IS IT AN "R" ?
        BNE     23$             ;NO? GO TO 23$
        DEC     BYTCNT          ;SUBTRACT 1 FROM THE BYTE COUNT
        BR      21$             ;CLEAN UP AND GO...
23$:    CMPB    #106,R0         ;IS IT A "F"?
        BNE     24$             ;NO? GO TO 24$
        MOV     #1,EOIFLG       ;SET THE EOI FLAG TO IGNORE EOI'S
        TST     OFFLAG          ;ON OR OFF THE EOIFLG
        BEQ     21$             ;IF OFFFLAG ISN'T SET, CLEAN UP AND GO
        CLR     OFFLAG          ;ELSE, CLEAR THE OFFLAG
        CLR     EOIFLG          ;CLEAR THE EOI FLAG
        BR      21$             ;CLEAN UP AND GO
24$:    CMPB    #127,R0         ;IS IT A "W" ??
        BNE     21$             ;NO??? UNDEFINED OPERATION, SO RETURN
        MOV     #1,WAITFL       ;SET THE WAIT FLAG
        TST     OFFLAG          ;IS THE OFF FLAG SET?
        BEQ     21$             ;GO TO 21$ IF NOT
        CLR     WAITFL          ;STOP WAIT FOR NRFD
        CLR     OFFLAG          ;AND OFF FLAG
        BR      21$             ;AND DONE
        .DSABL  LSB             ;NO MORE LOCAL SYMBOLS
;
;   SUBROUTINE GETBYT
GETBYT: MOV     PODB,R0         ;GET POINTER TO ODB INTO R0
        INC     PODB            ;UPDATE PODB
        MOVB    (R0),R0         ;GOT BYTE IN R0 NOW
        BNE     DONE            ;DONE IF IT'S NOT A ZERO
        JSR     PC,FPODB        ;IF ZERO, WE NEED A NEW PODB, SO GET IT
        TST     DONFLG          ;SEE IF NO MORE PODB'S TO BE HAD
        BEQ     GETBYT          ;IF WE GOT A GOOD PODB, THEN GET A BYTE
DONE:   RTS     PC              ;RETURN
;
;   SUBROUTINE FPODB... FETCH POINTER TO OPERATION DEFINITION BLOCK
FPODB:  MOV     PPDB,R0         ;GET POINTER TO PROCEDURE DEFINITION BLOCK
        ADD     #2,PPDB         ;UPDATE PPDB
        MOV     (R0),@#PODB     ;GET NEW PODB FROM PDB
        BNE     DONE1           ;  IF NOT ZERO, WE'RE DONE
        JSR     PC,FPPDB        ;IF ZERO WE NEED A NEW PPDB, SO GET IT.
        TST     DONFLG          ;BUT ARE THERE ANY MORE?
        BNE     FPODB           ;IF O.K., THEN GET THE PODB WE STARTED FOR
DONE1:  RTS     PC              ;DONE, SO RETURN
;
;   SUBROUTINE FPPDB... FETCH POINTER TO PROCEDURE DEFINITION BLOCK (WHEW!)
FPPDB:  INC     DONFLG          ;INITIALIZE DONE FLAG TO 1
        TST     NARG            ;SEE IF ANY MORE PPDB'S LEFT TO GET
        BEQ     DONE2           ;GO TO DONE2 IF NONE LEFT TO GET
        CLR     DONFLG          ;IF MORE, THEN CLEAR THE DONE FLAG
        MOV     (R5)+,PPDB      ;GET THE PPDB FROM THE FORTRAN CALLING THING
        DEC     NARG            ;THERE'S OBVIOUSLY ONE LESS NOW.
DONE2:  RTS     PC              ;RETURN.
;
```

```
;   THIS JUNK IS ALL THE VARIOUS POINTERS,FLAGS, BELLS AND WHISTLES THE
;   SUBROUTINES NEED TO BE HAPPY.......
BYTCNT: .WORD   0               ;THE BYTE COUNT FOR INPUT OF DATA
CKSUMR: .WORD   0               ;THE CHECKSUM READ FROM THE 7912AD
CKSUMF: .WORD   0               ;THE CHECKSUM COMPUTED DURING INPUT
OFFLAG: .WORD   0               ;A FLAG TO TELL CIF TO TURN OFF SOMETHING
OPFLAG: .WORD   0               ;A FLAG TO TELL CIF TO DO AN OPERATION
LAF:    .WORD   0               ;SET WHEN A LISTENER EXISTS SOMEWHERE
OUTWRD: .WORD   0               ;THIS WORD IS SENT TO THE DR11C FOR OUTPUT
DWA:    .WORD   0               ;THIS IS A POINTER TO THE INPUT DATA ARRAY
DONFLG: .WORD   0               ;THIS IS ON WHEN NO MORE BYTES TO PROCESS
NARG:   .WORD   0               ;THIS CONTAINS THE NUMBER OF PPDB'S LEFT TO DO
PODB:   .WORD   0               ;POINTER TO OPERATION DEFINITION BLOCK
PPDB:   .WORD   0               ;POINTER TO PROCEDURE DEFINITION BLOCK
ENDDEL: .WORD   0               ;THIS CONTAINS THE END DELIMITER TO BE USED
EOIFLG: .WORD   0               ;THIS IS SET WHEN EOI IS TO BE IGNORED
                                ;ON INPUT
WAITFL: .WORD   0               ;THIS FLAG IS SET TO STOP THE TIMOUT OF
                                ;TALKIT WHEN NRFD IS SLOW GOING HIGH
        .END
```

The next level of the software is the device-dependent level, consisting of some data blocks in order to make FORTRAN arrays accessible to the low level software, the PDBs, and the ODBs. The following four packages are overlaid by the system linker program with FORTRAN named common blocks. In this way, the entry points into these common blocks become globally named for use in the PDBs and ODBs. The source code follows:

```
FORTRAN:        COMMON /NUMSA/ NUMSA(4)

MACRO:
        .TITLE  NUMSA
        .GLOBL  NUMSA
        .PSECT  NUMSA,RW,D,GBL,REL,OVR
NUMSA:  .WORD   0
        .END


FORTRAN:        COMMON /VSCALF/ YSCALS(20)

MACRO:
        .TITLE  VSCALF
        .GLOBL  VSCALF
        .PSECT  VSCALF,RW,D,GBL,REL,OVR
VSCALF: .WORD   0
        .END


FORTRAN:        COMMON / HSCALF / HSCALS(20)

MACRO:

        .TITLE  HSCALF
        .GLOBL  HSCALF
```

```
        .PSECT  HSCALF,RW,D,GBL,REL,OVR
HSCALF: .WORD   0
        .END


FORTRAN:        COMMON /DATA/ DATA(512)

MACRO:

        .TITLE  DATA
        .GLOBL  DATA
        .PSECT  DATA,RW,D,GBL,REL,OVR
DATA:   .WORD   0
        .END
```

The next series is a list of operation definition blocks (ODBs) that are used to drive the various functions of the 7912AD digitizer. While not all listed here, one was developed for every possible function of the instrument, so that for future development of PDBs, most of the work has already been done. The source code follows:

```
        .TITLE  SEMIGO          ;sends terminator and end of message to 7912
        .GLOBL  SEMIGO
SEMIGO: .ASCIZ  /_W.;~./        ;SET WAIT FLAG, EOI,SEND ";", CANCEL EOI
        .EVEN
        .END

        .TITLE  NOWAIT          ;sets flag not to wait for handshake signal
        .GLOBL  NOWAIT          ;this is used for setting up to do input
NOWAIT: .ASCIZ  /~_W/           ;from 7912
        .EVEN
        .END

        .TITLE  RATC            ;command to read average-to-center data from
        .GLOBL  RATC            ; 7912AD
RATC:   .ASCIZ  /REA ATC.;~./
        .EVEN
        .END

        .TITLE  RSCF            ;READ SCALE FACTORS
        .GLOBL  RSCF,VSCALF,HSCALF
RSCF:   .ASCII  /_D/            ;LOAD DWA
        .WORD   VSCALF          ;WITH ADDRESS OF VERTICAL SCALE FACTOR ARRAY
        .ASCII  /_E;/           ;LOAD END DELIMITER
        .BYTE   377             ;WITH 377 IN HIGH, ";" IN LOW BYTE
        .ASCII  /<~_I/          ;LISTEN TO THE ";"
        .ASCII  /_D/            ;LOAD DWA
        .WORD   HSCALF          ;ADDRESS OF HORIZONTAL SCALE FACTOR ARRAY
        .ASCIZ  /_F<~_F/        ;DISABLE EOI EFFECT, LISTEN TIL ";", AND
                                ;RE-ENABLE EOI RESPONSE
        .EVEN
        .END
```

```
        .TITLE  SEMICO              ;command to send intra-message delimiter
        .GLOBL  SEMICO
SEMICO: .ASCIZ  /;/
        .EVEN
        .END

        .TITLE  DIGDAT              ;command to digitize data
        .GLOBL  DIGDAT
DIGDAT: .ASCIZ  /DIG DAT;/
        .EVEN
        .END

        .TITLE  UNTL                ;command to untalk-unlisten 7912AD
        .GLOBL  UNTL
UNTL:   .ASCIZ  /$_?_?~$/
        .EVEN
        .END

        .TITLE  ENDIT               ;command to terminate message to 7912
        .GLOBL  ENDIT
ENDIT:  .ASCIZ  /.;~./
        .EVEN
        .END

        .TITLE MAKLSN               ;command to make 7912 a listener
        .GLOBL  MAKLSN
MAKLSN: .ASCIZ/>!$  ~$/
        .EVEN
        .END

        .TITLE  MODDIG              ;command to put 7912 in digital mode
        .GLOBL  MODDIG
MODDIG: .ASCIZ  /MOD DIG;/
        .EVEN
        .END

        .TITLE  ATC                 ;command to 7912 to average-to-center
        .GLOBL  ATC                 ;data held in its memory
ATC:    .ASCIZ  /ATC;/
        .EVEN
        .END

        .TITLE  MAKTLK              ;command to make 7912 a talker
        .GLOBL  MAKTLK
MAKTLK: .ASCIZ  /!$@ ~$/
        .EVEN
        .END

        .TITLE  LLO                 ;command to 7912 to lock out local
        .GLOBL  LLO                 ;controls of instrument. Used to
LLO:    .ASCII  /$/                 ;prevent pushbutton-happy users from
        .BYTE   21                  ;interfering with computer control
        .ASCIZ  /~$/
        .EVEN
```

```
        .END

        .TITLE  GTL                     ;command to 7912 to go to
        .GLOBL  GTL                     ;local control
GTL:    .ASCII  /$/
        .BYTE   1
        .ASCIZ  /~$/
        .EVEN
        .END

        .TITLE  MODTV                   ;command to put 7912 into T.V. mode
        .GLOBL  MODTV
MODTV:  .ASCIZ  /MOD TV;/
        .EVEN
        .END

        .TITLE  GRATOF                  ;command to turn off graticule on 7912
        .GLOBL  GRATOF
GRATOF: .ASCIZ  /GRAT OFF;/
        .EVEN
        .END

        .TITLE  REDSA                   ;command to send signal-averaged data
        .GLOBL  REDSA                   ;from 7912 to computer
REDSA:  .ASCIZ  /REA SA.;~./
        .EVEN
        .END

        .TITLE  RBB                     ;command to read "block binary" data
                                        ;from 7912 to computer. See user's
                                        ;manual for data format.
        .GLOBL  RBB,ENDDEL,BYTCNT,CKSUMR,DATA
;
RBB:    .ASCII  /_D/            ;LOAD DWA
        .WORD   177777          ;WITH NOSTOR FLAG
        .ASCII  /_E/            ;LOAD END DELIMITER
        .ASCII  /%/             ;WITH % IN LOWBYTE
        .BYTE   177             ;AND FILL UPPER BYTE(NOT USED)
        .ASCII  /!</            ;LISTEN UNTIL INPUT DELIMITER
        .ASCII  /_D/            ;LOAD DWA
        .WORD   BYTCNT          ;WITH ADDRESS OF BYTECOUNT VARIABLE
        .ASCII  /_E/            ;LOAD END DELIMITER
        .WORD   177777          ;WITH READ TO BYTECOUNT FLAG
        .ASCII  /_B/            ;LOAD BYTECOUNT
        .BYTE   2               ;WITH 2
        .ASCII  /</             ;LISTEN FOR TWO BYTES
        .ASCII  /_R_D/          ;DECREMENT BYTECOUNT,THEN LOAD DWA
        .WORD   DATA            ;WITH ADDRESS OF DATA ARRAY
        .ASCII  /!</            ;AND READ TO BYTECOUNT
        .ASCII  /_D/            ;LOAD DWA
        .WORD   CKSUMR          ;WITH ADDRESS OF CHECKSUM READ
        .ASCII  /_E/            ;LOAD END DELIMITER
        .WORD   177777          ;WITH FLAG TO READ TO BYTECOUNT
        .ASCII  /_B/            ;LOAD BYTECOUNT
```

```
        .BYTE   1               ;WITH 1
        .ASCII  /</             ;READ 1 BYTE TO CKSUMR
        .ASCII  /_F/            ;SET FLAG TO IGNORE EOI ON INPUT
        .ASCII  /_D/            ;LOAD DWA
        .WORD   177777          ;WITH FLAG TO NOSTORE
        .ASCII  /_E;/           ;LOAD END DELIMITER WITH ";"
        .BYTE   377             ;UPPER BYTE(NOT USED)
        .ASCII  /</             ;INPUT TO DELIMITER,IGNORE EOI
        .ASCIZ  /~_F/           ;TURN OFF EOI FLAG
        .EVEN
        .END

        .TITLE  REASC1          ;routine to read scale factors from first
        .GLOBL  REASC1          ;plug in. If dual trace, returns upper trace
REASC1: .ASCIZ  /REA SC1.;~./   ;only
        .EVEN
        .END

        .TITLE  DIGSA           ;command to digitize ans signal average
        .GLOBL  DIGSA,NUMSA     ;"numsa" number of sequential transients.
DIGSA:  .ASCIZ  /DIG SA, /      ;see user's manual for format of numsa
        .EVEN
        .END

        .TITLE  OFFLIN          ; turns off REN line to make all IEEE devices
        .GLOBL  OFFLIN          ;go off line
OFFLIN: .ASCIZ /~!~$~.;/
        .EVEN
        .END
```

The final section of device dependent code consists of the PDBs. At the time of this writing, relatively few were actually written and used. The code for these follows:

```
;*************************************************************************
;* PDB to get the scale factors from the first set of plug-ins. Both a   *
;*    horizontal and vertical scale factor are returned.                 *
;*************************************************************************

        .TITLE  GETSCF
        .GLOBL  GETSCF,MAKLSN,UNTL,MAKTLK,FNDIT,LLO,GTL,RSCF,REASC1,OFFLIN
GETSCF: .WORD   MAKLSN          ;MAKE 7912 A LISTENER
        .WORD   LLO             ;DISABLE USER INTERFERENCE
        .WORD   REASC1          ;SEND COMMAND TO READ SCALE FACTORS 1
        .WORD   UNTL            ;UNLISTEN 7912
        .WORD   MAKTLK          ;ALLOW THE 7912 TO TALK
        .WORD   RSCF            ;READ SCALE FACTORS
        .WORD   UNTL            ;UNTALK/UNLISTEN THE 7912
        .WORD   MAKLSN          ;LISTEN THE 7912
        .WORD   GTL             ;GO TO LOCAL MODE
        .WORD   UNTL
        .WORD   0               ;END OF PROCEDURE DEFINITION BLOCK
        .END
```

```
;*************************************************************************
;* PDB to digitize the number of transient pulses contained in "NUMSA"  *
;*    averaging the data to center of scan, and dump the data into the  *
;*    data array.                                                       *
;*************************************************************************


        .TITLE  DUMPSA
        .GLOBL  DUMPSA,MAKLSN,MODDIG,DIGSA,REDSA,SEMIGO
        .GLOBL  UNTL,MAKTLK,RBB,ENDIT,LLO,GTL,NUMSA,NOWAIT
DUMPSA: .WORD   MAKLSN          ;MAKE 7912 A LISTENER
        .WORD   LLO             ;DISABLE USER INTERFERENCE
        .WORD   MODDIG          ;GO TO DIGITAL MODE
        .WORD   DIGSA           ;DIGITIZE DATA ON TRIGGER
        .WORD   NUMSA           ;ADDRESS OF THE ENCODED NUMBER OF SCANS
        .WORD   SEMIGO          ;A SEMICOLON AND GO TO FOLLOW THE NUMBER
        .WORD   REDSA           ;READ THE SA DATA
        .WORD   NOWAIT          ;UNDO THE WAIT FLAG SET BY SEMIGO
        .WORD   UNTL            ;UNTALK/UNLISTEN THE 7912
        .WORD   MAKTLK          ;MAKE THE 7912 A TALKER
        .WORD   RBB             ;READ IN THE BLOCK BINARY ATC DATA
        .WORD   UNTL            ;UNTALK THE 7912
        .WORD   MAKLSN          ;MAKE 7912 LISTENER
        .WORD   GTL             ;GO TO LOCAL MODE
        .WORD   UNTL
        .WORD   0               ;END OF PROCEDURE DEFINITION BLOCK
        .END

;*************************************************************************
;*  PDB to digitize a single transient, average it to center of sweep,  *
;*    and dump it to the data array.                                     *
;*************************************************************************

        .TITLE  DUMP
        .GLOBL  DUMP,MAKLSN,MODDIG,DIGDAT,ATC,RATC,NOWAIT
        .GLOBL  UNTL,MAKTLK,RBB,ENDIT,LLO,GTL,OFFLIN,SEMIGO
DUMP:   .WORD   MAKLSN          ;MAKE 7912 A LISTENER
        .WORD   LLO             ;DISABLE USER INTERFERENCE
        .WORD   MODDIG          ;GO TO DIGITAL MODE
        .WORD   DIGDAT          ;DIGITIZE DATA ON TRIGGER
        .WORD   ATC             ;AVERAGE THE DATA TO CENTER
        .WORD   SEMIGO          ;EXECUTE AND SET WAIT FLAG
        .WORD   RATC            ;READ THE ATC DATA
        .WORD   NOWAIT          ;CLEAR WAIT FLAG
        .WORD   UNTL            ;UNTALK/UNLISTEN THE 7912
        .WORD   MAKTLK          ;MAKE THE 7912 A TALKER
        .WORD   RBB             ;READ IN THE BLOCK BINARY ATC DATA
        .WORD   UNTL            ;UNTALK THE 7912
        .WORD   MAKLSN
        .WORD   GTL
        .WORD   UNTL
        .WORD   0               ;END OF PROCEDURE DEFINITION BLOCK
        .END
```

IV. THE USER SOFTWARE SYSTEMS

At this point it is suitable to present the user level software packages that the two experiments use. There are routines that are common to both programs, and those that are different. To minimize the amount of text, where differences are minor, only one version of the routine will be given, and differences noted. If sufficient differences exist, then both versions of a routine will be presented. While both follow approximately the same flow, differences in hardware have forced certain software differences. The system that supports the multiphoton work is equipped with a raster scan display processor and graphics display unit. Therefore, the graphics package for the multiphoton program utilizes this hardware. For the lifetime program, a Tektronix 4014 look-alike terminal was assumed to be the graphics output device, and all graphics code written for it. As mentioned previously, the multiphoton program also operated a shutter, two stepper motors, and a high voltage switch, so that there are differences in the programs due to this. Finally, in the lifetime program, it was deemed necessary to be able to do some on-line curve fitting to an exponential function, so that in this program a means was implemented to store all relevant data about the current context of the program in a disc file, and then start up a user fitting program which would then access that file in order to do this additional processing. While this second program is running, the first has been completely halted and no longer exists as a process in the system. When the user analysis program has completed, the lifetime program is again automatically re-started. A program may make a system call to detect being started in this fashion. If automatic start is detected, the lifetime program reads the disc context file, restores the program context, and continues on as if the intervening program had never been executed. This so called "chaining" of programs provides a way of implementing multi-tasking and process-to-process communications in a simple way.

The programs are written in the top-down structure of modern structured programming, even though FORTRAN is not a structured language. In addition, the programs are microprogrammable, i.e., they may have sequences of operations defined, and then executed repetitively as microprograms. In this way, as day to day use of the program changes, only the user defined microprograms need to change, not the main routines. In order to accomplish this, a particular architecture was used, and is shown in Figure 7. Both of the programs have this structure. They differ primarily in data structures and extra features caused by hardware differences. The operation of the software was designed to simulate the classic Von Neuman cycle that most computers use:

1. Fetch an instruction from memory
2. Increment the program counter to point to the next one in memory
3. Decode the instruction
4. Fetch any additional required operands, updating the program counter as necessary
5. Execute the instruction

When this cycle is repeated, a program consisting of a list of instructions and operands stored in memory is executed. It is due to this method of operation that computers do not have to be re-wired for every new program, and

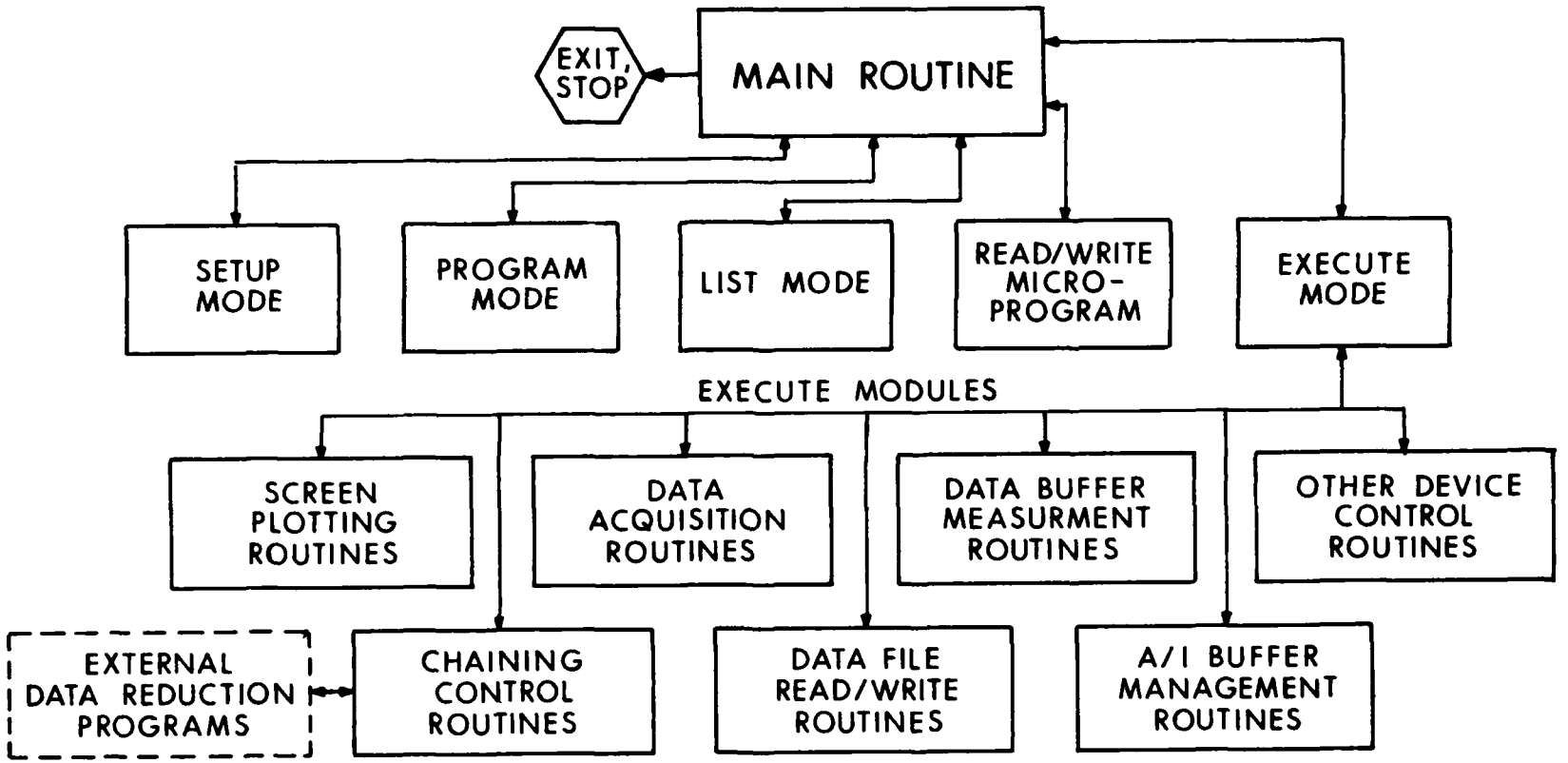


Figure 7. The Architecture of the User Level Programs

the user programs described here do not have to be re-written for every new sequence of operations they are to perform. The programs all have the same parts: the setup section, a programming section, an execution section, and a main routine. The programs both have an instruction array to hold the instructions to be executed, both have a program counter to point to the next item in the instruction array to be fetched, and both have manual and automatic modes of operation. In the automatic mode, the user begins in the main routine, and selects to go to the programming mode. In this mode, a series of menus are displayed listing the options available. The user selects those operations desired, and the programming mode places the corresponding opcodes and arguments in the instruction array. When this microprogram is complete, and the sequence of operations that the user desires is defined, the user specifies that a return to the main routine is desired. In the main routine, the user may review his microprogram via a listing option, may store the microprogram on disc for future use, may read in a microprogram from disk that was stored previously, or may go to the execution mode. In the execution mode, the user is queried for an execution count (N). The microprogram will then be completely executed N times, after which a return to the main routine is done. The manual mode is a degenerate case of the automatic mode. When the manual mode is selected, a flag is set, and the execution count set to 1. Next, automatic entry is done to the programming mode. In this mode, whenever a complete instruction is entered into the instruction array, the manual flag is tested. If set, a return is made to the main routine where the flag is again tested. If set, an automatic entry is made to the execution routine and the microprogram (consisting of the single instruction) executed. Since the execution count was 1, a return to main routine is done, the manual flag tested, and the automatic entry into programming mode done again. In this way, the user can give single commands which are executed immediately. This mode is useful in experiments where considerable "poking around" must be done prior to being able to define a set sequence of operations for automatic data acquisition and control. In keeping with the top-down approach in program design, the same approach will be used in program description. The first module, then, to be described is the main control module. In order to begin, the data structures will be defined.

The following lines merely override any implicit variable types that FORTRAN defaults to:

```
INTEGER DATA,AUTINC,SELECT,MANUAL
LOGICAL*1 FILE1,FILE2,FILE3,YSCALS,HSCALS,NUMSA
LOGICAL*1 IDINF1,IDINF2,FILE4
```

Next, the common blocks or global segments of the data structures:

COMMON /DATA/ DATA(512)

This 512 integer array is used by the PDBs to accept the data dumped from the 7912AD digitizer. The range of the data depends on the mode that is used to dump it. For a single scan, the range is 0 - 511. For multiple scans, it is this range multiplied by the number of scans.

COMMON /VSCALF/ YSCALS(20)

This 20 byte array is used by the PDBs to receive the vertical scale factors sent by the 7912AD. Since the factors are sent in ASCII code, routines CVTSCF and SORTIT are used to convert to internal format floating point numbers for use in the program.

COMMON /HSCALF/ HSCALS(20)

This 20 byte array has the same function and description as that for VSCALF, except that it receives the horizontal scale factors.

COMMON /NUMSA/ NUMSA(4)

This 4 byte array is loaded with the ASCII code for the number of repetitive pulses to be digitized and summed within the 7912AD. The array is used by the PDBs to tell the CIF routine where to get the character string. The name stands for NUMber to Signal Average.

COMMON /BUFFA/ RADATA(512)

This 1024 word (2048 byte) array contains the 512 floating point data values of the symbolic data buffer "A". It is used as a general data accumulator.

COMMON /BUFFB/ RBDATA(512)

This corresponds to buffer "B", with the same description as BUFFA.

COMMON /BUFFD/ RDDATA(512)

This corresponds to buffer "D", same description as BUFFA.

COMMON /CNTRL/ YSCALE,XSCALE,NPULSE,MANUAL,IRFLN,ICOFLN
1,FILE1(16),FILE2(16),FILE3(16),FILE4(16),AUTINC

As there are numerous variables in this statement, they are broken out as follows:

YSCALE,XSCALE

These are the vertical and horizontal scale factors corresponding to VSCALF and HSCALF. They are floating point numbers, and represent volts/division and time/division respectively. There are 8 divisions on the screen of the 7912AD, corresponding to 512 units in the transmitted data. If, for instance, the time/div were 0.001 seconds/div, the time per "channel" would be 15.625 microseconds (.008/512).

NPULSE

This is an integer that contains the Number of PULSEs that the program is to signal average. This is different from NUMSA, in that NUMSA may only contain numbers that are powers of 2, i.e., 1, 2, 4, 8, 16, 32, or 64. Any numbers other than these are truncated internally to the 7912AD into a power

of two, and then that number of pulses digitized. In order to average a number higher than 64, or an odd number of pulses, that number must be broken into the powers of 2, and done in parts. NPULSE contains that actual number to be done, and must be less than 32768.

MANUAL

This is the flag that indicates whether the program is in automatic (microprogramming) or manual (inter-pretive) mode. MANUAL = 1 indicates manual mode.

IRFLN

This is a flag that if set to 1 will cause the program to request file names before any file accesses are performed. If set to zero, the file name number indicated in the instruction opcode is used to indicate which internally stored file name string is to be used.

ICOFLN

This is a flag that when set to 1 causes the file name used in a file access to be printed on user terminal after its use.

FILE1,FILE2,FILE3,FILE4

These 4 arrays may be loaded with ASCII strings representing file names by the user. The first 3 are used for reading and writing the data buffers A, B, and D to disc files, and the fourth for reading and writing the A/I buffer to disc files.

AUTINC

This flag controls the autoincrement feature of file access. Under this feature, file names are entered with the form "DEV:FILNAM.EXT", where DEV is an RT-11 device, FILNAM is a 1 to 6 letter file name, and EXT is a NUMERICAL (e.g. 003) extension. The file name is entered into one of the strings mentioned above. After each time the name is used to open a file for a read or write, the extension is incremented by 1 if AUTINC = 1. Otherwise, the filename is unchanged.

COMMON /ID/ IDINF1(72),IDINF2(72)

These byte arrays are used to store identification information about the data in the buffers. They are stored with the data arrays when a file is created, and loaded from a file when it is read.

COMMON /MUPROG/ INSTR(100)

This is the instruction array where the programming routine stores its opcodes, and from which the execution routine reads them. The array may be stored in a disc file, or loaded from a disc file.

COMMON /AI/ AIX,AIY,AIVALS(100),IPTR,AIX,AIY

These variables hold the most recent measurement taken by one of the data measurement routines (integral, amplitude, etc.).

AIVALS

This array is sequentially loaded with successive measurements, upon microprogram command. This buffer may be stored to or loaded from a disc file, and may be plotted on the screen by the data acquisition pro-grams, or made into hard copy by external programs.

IPTR

This is a pointer into the AIVALS array. It points to the next free element in the array.

COMMON /PLT/ SCALF,YMIN,YMAX

These variables are for use by the plotting routines. SCALF is the vertical scale factor, and YMIN,YMAX are the vertical data minimum and maximum values respectively.

COMMON /CURSOR/ IX1,Y1,IX2,Y2,IWAIT,IDFSE1,IDFSE2

IX1,Y1

For the graphics package that supports the MDP-3 display processor, two cursor positions are stored, in order to speed up the positioning of the cursor during multiple measurements. These variables are the X and Y values of the first.

IX2,Y2

These are the X and Y positions of the second stored cursor position.

IWAIT

This is a flag that determines whether or not the program will wait for the user to move the cursor from its stored position before using its X, Y values for input to the measurement routines. For data measurement sets where it is desirable to always perform the measurements on the same portion of the data, and cursor movement is unnecessary, this flag is set to zero, and no user interaction is necessary to perform the indicated measurements.

IDFSE1,IDFSE2

These are flags indicating that the two cursor positions have been loaded. It is used to insure that the user has put valid positions in storage before the IWAIT flag can be set not to allow user cursor movement.

COMMON /PLTLIM/ IFIRST,ILAST

These are the first and last channels in "X" that define the active data set range.

And now, the code for the main routine of the multiphoton program:

```
      PROGRAM MULTIP
      INTEGER DATA,AUTINC,SELECT,MANUAL
      LOGICAL*1 FILE1,FILE2,FILE3,YSCALS,HSCALS,NUMSA
      LOGICAL*1 IDINF1,IDINF2,FILE4
      COMMON /DATA/ DATA(512)
      COMMON /VSCALF/ YSCALS(20)
      COMMON /HSCALF/ HSCALS(20)
      COMMON /NUMSA/ NUMSA(4)
      COMMON /BUFFA/ RADATA(512)
      COMMON /BUFFB/ RBDATA(512)
      COMMON /BUFFD/ RDDATA(512)
      COMMON /CNTRL/ YSCALE,XSCALF,NPULSE,MANUAL,IRFLN,ICOFLN
      1,FILE1(16),FILE2(16),FILE3(16),FILE4(16),AUTINC
      COMMON /ID/ IDINF1(72),IDINF2(72)
      COMMON /MUPROG/ INSTR(100)
      COMMON /AI/ AIX,AIY,AIVALS(100),IPTR
      COMMON /PLT/ SCALF,YMIN,YMAX
      COMMON /CURSOR/ IX1,Y1,IX2,Y2,IWAIT,IDFSE1,IDFSE2
      COMMON /PLTLIM/ IFIRST,ILAST
      DATA NUMSA /4 * 0/,AUTINC/0/,IRFLN/1/,ICOFLN/0/
      DATA IPTR /1/,AIVALS /100 * 0.0/,IWAIT/1/,IDFSE1/0/,IDFSE2/0/
      DATA IFIRST /1/,ILAST/512/
      MANUAL = 0
C  Type the headers, main menu
      TYPE 100
7     TYPE 101
      TYPF 102
      TYPE 103
C  Get the selection, and got to that section of code
      ACCEPT 104,SELECT
      GO TO (1,2,3,4,5,6,9,8)SELECT
      GO TO 7
C  Go to setup mode
1     CALL SETUP
      GO TO 7
C  Go to microprogramming mode
2     CALL PROGRM
      IF(MANUAL.EQ.0)GO TO 7
      GO TO 4
C  Go to manual mode of operation
3     CALL GOTOMA
      GO TO 2
4     IF(MANUAL.NE.1)GO TO 10
      NUMREM = 1
      GO TO 11
C  Get execution count
10    TYPE 105,NUMREM
      ACCEPT 104,MUMREM
```

```
        IF(MUMREM.NE.0)NUMREM=MUMREM
C  Go to execution mode
11      CALL EXECUT(NUMREM)
        IF (MANUAL.EQ.0)GO TO 7
        GO TO 2
C  Save microprogram on disc
5       CALL OUTPRO
        GO TO 7
C  Go get a microprogram from disc
6       CALL INPROG
        GO TO 7
C  List current microprogram in memory
9       CALL LISTPR
        GO TO 7
8       CALL EXIT
C******************** FORMAT STATEMENTS FOR ROUTINE MAIN ***************


100     FORMAT(1X,
        1'THIS IS THE MULTI-PHOTON MEASUREMENT DATA ACQUISITION PROGRAM'/)
101     FORMAT(5X'MAIN SELECTION MENU: '//)
102     FORMAT(10X'1 = ENTER SETUP FOR SERIES MODE'/
        1,10X'2 = ENTER PROGRAMMING MODE'/
        2,10X'3 = GO TO MANUAL MODE'/
        3,10X'4 = EXECUTE MICROPROGRAM'/
        4,10X'5 = STORE MICROPROGRAM ON DISC'/
        5,10X'6 = READ IN MICROPROGRAM FROM DISC'/
        7,10X'7 = LIST MICROPROGRAM'/
        6,10X'8 = EXIT FROM PROGRAM'//)
103     FORMAT('$',12X,'SELECTION ? >')
104     FORMAT(I6)
105     FORMAT('$NUMBER OF REPETITIONS ? (',I4,') >')

C*******************************************************************************

        END
```

The data structures for the lifetime program include a few extras, and leave out some of those in the multiphoton program. Those that are left out require no explanation, and the extras follow:

NWTPLT

This flag is included in LIFTIM, since the control terminal and graphics output device are one and the same. In order to allow the user to observe the plots without printing on the screen, the program pauses after plotting, waiting for the user to type a carriage return, after which the plot is erased, and the normal terminal operation restored. Setting the NWTPLT flag to 1 causes this pause not to occur, allowing the microprogram to continue without user interaction. This is useful for viewing large numbers of data files in a "movie" type progression, while the user sits back and watches.

PC

While this is not a new variable in the programs, it is included in common in the lifetime program due to the chaining in use. In this way, if a chain call is made in the middle of a microprogram, the value of the PC is stored for the chain out, and restored during the chain-in so that the next logical microprogram step will be executed after the chain is completed.

The code for the main routine of the lifetime program follows:

```
      PROGRAM LIFTIM

      INTEGER DATA,AUTINC,SELECT,MANUAL,PC
      LOGICAL*1 FILE1,FILE2,FILE3,YSCALS,HSCALS,NUMSA
      LOGICAL*1 IDINF1,IDINF2,FILE4
      COMMON /DATA/ DATA(512)
      COMMON /VSCALF/ YSCALS(20)
      COMMON /HSCALF/ HSCALS(20)
      COMMON /NUMSA/ NUMSA(4)
      COMMON /BUFFA/ RADATA(512)
      COMMON /BUFFB/ RBDATA(512)
      COMMON /BUFFD/ RDDATA(512)
      COMMON /CNTRL/ YSCALE,XSCALE,NPULSE,MANUAL,IRFLN,ICOFLN
      1,FILE1(16),FILE2(16),FILE3(16),FILE4(16),AUTINC,NWTPLT
      COMMON /ID/ IDINF1(72),IDINF2(72)
      COMMON /MUPROG/ INSTR(100),PC
      COMMON /AI/ AIX,AIY,AIVALS(100),IPTR
      COMMON /PLT/ SCALF,YMIN,YMAX
      COMMON /CURSOR/ IX1,Y1,IX2,Y2,IWAIT,IDFSE1,IDFSE2
      COMMON /PLTLIM/ IFIRST,ILAST
      DATA NUMSA /4 * 0/,AUTINC/0/,IRFLN/1/,ICOFLN/0/
      DATA IPTR /1/,AIVALS /100 * 0.0/,NWTPLT/0/,IDFSE1/0/,IDFSE2/0/
      DATA IFIRST/1/,ILAST/512/

C     CODE TO DETECT CHAIN-IN AND SETUP TO RESUME WHERE LEFT OFF
      CALL RCHAIN(IFLAG,IVAR,0)
C     set terminal to VT-100 mode and clear screen
      CALL SETTRM
      CALL VTMODE
      CALL VTPAGE
      IF(IFLAG.NE.-1)GO TO 12
C     restore software context
      CALL RESTOR(NUMREM)
      ICHAIN = 1
      GO TO 11
C     NORMAL ENTRY CODE POINT
12    MANUAL = 0
      ICHAIN = 0
C     Put on headings, main select menu..
      TYPE 100
7     TYPE 101
      TYPE 102
      TYPE 103
      ACCEPT 104,SELECT
```

```
        GO TO (1,2,3,4,5,6,9,8)SELECT
        GO TO 7
1       CALL SETUP
        GO TO 7
2       CALL PROGRM
        IF(MANUAL.EQ.0)GO TO 7
        GO TO 4
3       CALL GOTOMA
        GO TO 2
4       IF(MANUAL.NE.1)GO TO 10
        NUMREM = 1
        GO TO 11
10      TYPE 105,NUMREM
        ACCEPT 104,MUMREM
        IF(MUMREM.NE.0)NUMREM = MUMREM
11      CALL EXECUT(NUMREM,ICHAIN)
        IF (MANUAL.EQ.0)GO TO 7
        GO TO 2
5       CALL OUTPRO
        GO TO 7
6       CALL INPROG
        GO TO 7
9       CALL LISTPR
        GO TO 7
8       CALL EXIT
C******************** FORMAT STATEMENTS FOR ROUTINE MAIN ***************


100     FORMAT(1X,
        1'THIS IS THE LIFETIME MEASUREMENT DATA ACQUISITION PROGRAM'/)
101     FORMAT(5X'MAIN SELECTION MENU: '//)
102     FORMAT(10X'1 = ENTER SETUP FOR SERIES MODE'/
        1,10X'2 = ENTER PROGRAMMING MODE'/
        2,10X'3 = GO TO MANUAL MODE'/
        3,10X'4 = EXECUTE MICROPROGRAM'/
        4,10X'5 = STORE MICROPROGRAM ON DISC'/
        5,10X'6 = READ IN MICROPROGRAM FROM DISC'/
        7,10X'7 = LIST MICROPROGRAM'/
        6,10X'8 = EXIT FROM PROGRAM'//)
103     FORMAT('$',12X,'SELECTION ? >')
104     FORMAT(I6)
105     FORMAT('$NUMBER OF REPETITIONS ?(',I3,') >')

C*****************************************************************************

        END

C*****************************************************************************
```

Probably the next logical level in the routines is the programming routines. These routines only place opcodes into the instruction array; they do not execute any real functions. In order to begin the description of the microprogramming, a list of the opcode assignments is given, and their

functions. The descriptions are all in capital letters since this is a copy of an on-line documentation file that is part of the software package. The file follows:

```
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$  OPCODE DEFINITION DOCUMENTATION FILE FOR THE PROGRAMS $
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
```

OPCODE	NUMBER ARGS	MEANING OF OPCODE
1	0	ERASE CONTENTS OF "A"
2	0	ERASE CONTENTS OF "B"
3	0	ERASE CONTENTS OF "D"
4	0	ERASE CONTENTS OF ALL BUFFERS
5	0	ACCUMULATE SINGLE PULSE TO "A"
6	0	ACCUMULATE SINGLE PULSE TO "B"
7	0	ACCUMULATE SINGLE PULSE TO "D"
8	1	ACCUMULATE MULTIPLE PULSES INTO "A", ARG1 = NUMBER OF PULSES TO ACCUMULATE
9	1	ACCUMULATE MULTIPLE PULSES INTO "B", ARG1 = NUMBER OF PULSES TO ACCUMULATE
10	1	ACCUMULATE MULTIPLE PULSES INTO "D", ARG1 = NUMBER OF PULSES TO ACCUMULATE
11	1	PLOT SINGLE DATA SET,ARG1 = BUFFER 1,2,3
12	2	PLOT 2 DATA SETS IN ARG1,ARG2... ARG1 = FIRST BUFFER, ARG2 = SECOND BUFFER
13	3	PLOT 3 DATA SETS IN ARG1,ARG2,ARG3. ARG1 = FIRST BUFFER,ARG2 = SECOND BUFFER, ARG3 = THIRD BUFFER
14	1	WRITE DATA IN BUFFER "A" TO FILE. ARG1 = FILNAME NUMBER 1,2, OR 3
15	1	WRITE "B" TO FILE. ARG1 = FILENAME NUMBER
16	1	WRITE "D" TO FILE. ARG1 = FILENAME NUMBER
17	1	READ DATA INTO BUFFER "A" FROM FILE. ARG1 = FILNAME NUMBER 1,2, OR 3.
18	1	READ "B" FROM FILE. ARG1 = FILENAME NUMBER
19	1	READ "D" FROM FILE. ARG1 = FILENAME NUMBER

20	0	WRITE AMPLITUDE/INTEGRAL DATA BUFFER TO FILE.
21	0	ERASE A/I DATA BUFFER
*22	1	MOVE BURNER +/- STEPS, ARG1 NUMBER OF STEPS.
*23	0	OPEN SHUTTER
*24	0	CLOSE SHUTTER
25	0	READ DATA INTO A/I BUFFER
26	0	TYPE OUT DATA IN A/I BUFFER ON SCREEN
27	0	COMPUTE A-B PUT IN D
28	1	INCREMENT FILNAME EXTENSION,ARG1 = FILENAME NUMBER
29	0	INTEGRATE BETWEEN CURSOR LIMITS
30	0	MOVE LAST MEASUREMENT TO A/I BUFFER, NEXT POSITION
31	0	MEASURE AMPLITUDES OF PULSES,FROM CURSOR.
32	1	MEASURE AMPLITUDES OF PULSES, FROM DATA SET. ARG1 = BUFFER NUMBER, CHANNEL NUMBER FROM CURSOR
33	0	WAIT FOR USER TO TYPE A CARRIAGE RETURN
34	0	ACQUIRE SCALE FACTORS ONLY
*35	1	MOVE DYE LASER GRATING. ARG1=#STEPS OF .012 NM
36	1	MEASURE AMPLITUDES QUICK AND DIRTY FROM MAX-MIN. ARG1 = BUFFER NUMBER
**37	0	CHAIN TO FITTER PROGRAM
38	0	PLOT CONTENTS OF A/I BUFFER ON SCREEN
*39	0	TURN ON HIGH VOLTAGE TO ION COLLECTOR
*40	0	TURN OFF HIGH VOLTAGE TO ION COLLECTOR

* INDICATES THAT THIS OPCODE IS FOR MULTIPHOTON PROGRAM ONLY
** INDICATES THAT THIS OPCODE IS FOR LIFETIME PROGRAM ONLY

$$$

AUTOINCREMENT : WHEN SET, A FILENAME EXTENSION IS INCREMENTED EACH TIME AFTER IT IS USED.

REQUEST FILENAMES: WHEN SET, THE PROGRAM WILL ASK FOR THE FILENAME TO BE USED BEFORE THE FILE OPERATION

MANUAL: WHEN SET, THE PROGRAM IMMEDIATELY ENTERS THE PROGRAMMING MODE, AND THEN EXECUTES THE OPERATION SELECTED IMMEDIATELY, AND RETURNS TO PROGRAMMING MODE.

ECHO FILE NAMES : WHEN SET THE ACTUAL FILE NAME USED IS ECHOED ON FILE ACCESS

CHAIN TO FITTER OPCODE: THIS ACTUALLY CAUSES ALL RELEVANT INFORMATION ABOUT THE PROGRAM TO BE SAVED IN A CHAIN FILE, AND A NEW PROGRAM TO BE LOADED INTO CORE AND RUN. THE LIFTIM PROGRAM IS OVERWRITTEN. WHEN THE LIFTIM PROGRAM IS CHAINED INTO, THE CHAIN FILE IS READ AND RESTORES THE PROGRAM TO ITS ORIGINAL STATE, ALL VARIABLES INTACT.

INTERRUPT EXECUTION: THE PROGRAM MAY BE INTERRUPTED AT ANY POINT EXCEPT DURING ACTUAL DATA ACQUISITION BY THE 7912 DIGITIZER BY TYPING TWO CONSECUTIVE CONTROL C'S. THEN WHEN "REE" IS TYPED THE PROGRAM WILL RESTART, WITH ALL BUFFERS AND OPTIONS STILL IN EFFECT.

ABORT PREVIOUS A/I MEASUREMENT: THIS MOVES THE CURRENT A/I BUFFER POINTER BACK AND ERASES THE VALUES FOUND THERE. THIS IS USEFUL TO CANCEL A PART OF A DATA SET WHERE THE INSTRUMENTS WENT WEST DURING A DATA RUN.(FOUND IN SETUP MODE)

SELECT DISPLAY SUBSET: THIS SELECTS THE SUBRANGE OF DATA TO BE DISPLAYED AND OPERATED UPON BY THE DATA MEASUREMENT/CALCULATION OPERATIONS. THIS IS USEFUL TO ELIMINATE SECTIONS OF THE INPUT DATA SET WHERE NOISE WOULD ONLY CONFUSE MATTERS.(FOUND IN SETUP MODE)

OUICK-AND-DIRTY AMPLITUDES: THIS AVERAGES THE FIRST 15 CHANNELS OF THE SELECTED DATA SET IN THE SELECTED SUBRANGE AND ASSIGNS THIS TO BE ZERO, AND THEN CALCULATES THE MAX IN THE DATA SUBRANGE. AMPLITUDE = MAX - ZERO LEVEL.

********** END OF OPCODES DOCUMENTATION FILE **********

These are the actual numbers that are written into the instruction array by the programming section of the programs. Since the opcodes are the same for both programs, microprograms from one program may be executed by the other. Unidentified opcodes do, however, cause an error message to be printed to the user.

In order to microprogram (or do manual operations), the opcodes must be

entered into the instruction array. To do this, the program counter is set to 1, meaning the first element in the array, the array is cleared to zeros, and the opcodes placed in the array. The PC is used to tell the programming routine which array element to place the numerical code into, after which the PC is incremented. In order to make the user interface as painless as possible, menus are used to select categories of functions, and sub-menus to select actual functions where needed. This was found to require the least user sophistication in order to accomplish the user goal. As a result of this simplicity, and the power afforded by microprogramming, the complexity and sophistication of the user microprogramming matches the expertise of the user at all times. The code for the programming section of the multiphoton program follows:

```
        SUBROUTINE PROGRM
        COMMON /CNTRL/ YSCALE,XSCALE,NPULSE,MANUAL,IRFLN,ICOFLN
        1,FILE1(16),FILE2(16),FILE3(16),FILE4(16),AUTINC
        COMMON /MUPROG/ INSTR(100)
        INTEGER PC,SELECT
        DATA IY/'Y'/, IA/'A'/ , IB/'B'/ , ID/'D'/

        ICMM = 157.48031
        DO 13 I=1,100
        INSTR(I)=0
13      CONTINUE
        PC = 1
        TYPE 100
1       TYPE 101
2       TYPE 102
        ACCEPT 103,SELECT
        GO TO (3,4,5,6,7,8,9,10,11)SELECT
        GO TO 1
C  ERASE BUFFERS OPCODES
3       TYPE 104
        ACCEPT 116,ICH
        IF (ICH.NE.IY)GO TO 31
        INSTR(PC)=4
        GO TO 32
31      CALL GETBUF(ICH)
        INSTR(PC)=ICH
32      PC=PC+1
        GO TO 2
        RETURN
C  ACCUMULATE SINGLE PULSE OPCODES
4       TYPE 117
45      TYPE 102
        ACCEPT 103,ICH
        GO TO (41,42,43,12)ICH
        GO TO 4
41      CALL GETBUF(ICH)
        INSTR(PC)=ICH+4
        GO TO 44
C   ACCUMULATE MULTIPLE PULSES OPCODES
42      CALL GETBUF(ICH)
        INSTR(PC)=ICH + 7
```

```
        PC=PC+1
        TYPE 106
        ACCEPT 103,INSTR(PC)
        GO TO 44
C  ACCUMULATE SCALE FACTORS ONLY OPCODE
43      INSTR(PC)=34
44      PC=PC+1
        GO TO 45
        RETURN
C  PLOT DATA OPCODES AND ARGS
5       TYPE 107
        ACCEPT 103,ICH
        IF(ICH.LT.1.OR.ICH.GT.3)GO TO 5
        INSTR(PC)=ICH + 10
        PC = PC + 1
        DO 51 I=1,ICH
        CALL GETBUF(IVAL)
        INSTR(PC)=IVAL
        PC=PC+1
51      CONTINUE
        GO TO 12
        RETURN
C  FILE OPERATIONS SECTION
6       TYPE 108
61      TYPE 102
        ACCEPT 103,ICH
        GO TO (62,62,63,12)ICH
        GO TO 6
C  WRITE/READ DATA BUFFER OPCODES
62      CALL GETBUF(IVAL)
        IF(ICH.EQ.1)INSTR(PC)=IVAL+13
        IF(ICH.EQ.2)INSTR(PC)=IVAL+16
        PC=PC+1
        GO TO 64
C  INCREMENT EXTENSION OPCODE
63      INSTR(PC)=28
        PC=PC+1
C  GET ARG SECTION
64      CALL GETFIL(IVAL)
        INSTR(PC)=IVAL
        PC=PC+1
        GO TO 61
C  MISCELLANEOUS OPERATIONS  SECTION
7       TYPE 112
71      TYPE 102
        ACCEPT 103,ICH
        GO TO (72,73,74,75,76,79,791,792,12)ICH
        GO TO 7
C  MOVE BURNER BY MM
72      TYPE 113
        ACCEPT 115,RMM
        ICNT=RMM * ICMM + 0.5
        GO TO 77
C  MOVE BURNER BY COUNTS
```

```
73      TYPE 114
        ACCEPT 103,ICNT
C  ASSEMBLE BURNER MOVE INSTRUCTION
77      INSTR(PC)=22
        PC=PC+1
        INSTR(PC)=ICNT
        GO TO 78
C  OPEN SHUTTER OPCODE
74      INSTR(PC)=23
        GO TO 78
C  CLOSE SHUTTER OPCODE
75      INSTR(PC)=24
        GO TO 78
C  WAIT FOR CR OPCODE
76      INSTR(PC)=33
C  GENERAL RETURN CODE
78      PC=PC+1
        GO TO 71
C  MOVE LASER OPCODE
79      INSTR(PC) = 35
        PC = PC + 1
        TYPE 118
        ACCEPT 103, INSTR(PC)
        GO TO 78
C  ION COLLECTOR VOLTAGE CONTROL OPCODES
791     INSTR(PC)=39
        GO TO 78
792     INSTR(PC)=40
        GO TO 78
C  AMPLITUDE/INTEGRAL BUFFER OPERATIONS
8       TYPE 110
81      TYPE 102
        ACCEPT 103,ICH
        GO TO (82,83,84,85,87,12)ICH
        GO TO 8
C  ERASE A/I OPCODE
82      INSTR(PC)=21
        GO TO 86
C  TYPE OUT CONTENTS OF A/I OPCODE
83      INSTR(PC)=26
        GO TO 86
C  WRITE A/I TO FILE OPCODE
84      INSTR(PC)=20
        GO TO 86
C  READ FILE INTO A/I OPCODE
85      INSTR(PC)=25
        GO TO 86
C  PLOT A/I BUFFER OPCODE
87      INSTR(PC)=38
C  GENERAL RETURN CODE
86      PC=PC+1
        GO TO 81

C  DATA MEASUREMENT/CALCULATIONS CODES
```

```
9       TYPE 111
91      TYPE 102
        ACCEPT 103,ICH
        GO TO (92,93,93,93,97,98,12)ICH
        GO TO 9
C  OPCODE FOR D=A-B OPERATION
92      INSTR(PC)=27
        GO TO 96
C  CODE TO GET BUFFER AND PLOT IT FOR AMPLITUDE MEASURE
93      CALL GETBUF(IVAL)
        INSTR(PC)=11
        PC=PC+1
        INSTR(PC)=IVAL
        PC=PC+1
        IF(ICH.EQ.3)GO TO 94
        IF(ICH.EQ.4)GO TO 95
C  OPCODE FOR AMPLITUDE FROM CURSOR POSITION
        INSTR(PC)=31
        GO TO 96
C  OPCODE FOR AMPLITUDE FROM DATA SET
94      INSTR(PC)=32
        PC=PC+1
        INSTR(PC)=IVAL
        GO TO 96
C  OPCODE FOR INTEGRATE BETWEEN LIMITS
95      INSTR(PC)=29
        PC=PC+1
        INSTR(PC)=IVAL
96      PC=PC+1
        GO TO 91
C  OPCODE FOR MOVE DATUM TO A/I BUFFER
97      INSTR(PC)=30
        GO TO 96
C  OPCODE FOR QUICK-N-DIRTY AMPLITUDE
98      INSTR(PC)=36
        PC=PC+1
        CALL GETBUF(INSTR(PC))
        GO TO 96
C  GO TO AUTOMATIC MODE
10      MANUAL = 0
        RETURN
12      IF(MANUAL.EQ.0)GO TO 1
11      RETURN

C******************* FORMAT STATEMENTS FOR SUBROUTINE PROGRM ***************
100     FORMAT(1X'******** PROGRAMMING MODE *********'//)
101     FORMAT(1X'PROGRAM SELECTION MENU:'/
        1,10X'1 = ERASE DATA BUFFER(S)'/
        2,10X'2 = DIGITIZE DATA'/
        3,10X'3 = PLOT DATA ACCUMULATED'/
        4,10X'4 = DO FILE OPERATIONS'/
        5,10X'5 = GO TO MISCELLANEOUS HANDLER MENU '/
        6,10X'6 = DO AMPLITUDE/INTEGRAL BUFFER OPERATIONS'/
        7,10X'7 = DO DATA MEASUREMENT/CALCULATIONS'/
```

```
        8,10X'8 = GO TO AUTOMATIC MODE FROM MANUAL'/
        9,10X'9 = RETURN TO MAIN SELECT MENU'//)
102     FORMAT('$',9X'SELECTION ? >')
103     FORMAT(I5)
104     FORMAT('$ALL BUFFERS ? (Y OR N)>')
106     FORMAT('$NUMBER OF PULSES ? (1 - 32767) >')
107     FORMAT('$NUMBER OF PLOTS ON SCREEN ? (1-3) >')
108     FORMAT(1X'FILE HANDLER SELECTIONS:'//
        1,10X'1 = WRITE DATA BUFFER TO FILE'/
        2,10X'2 = READ DATA FROM FILE TO DATA BUFFER'/
        3,10X'3 = INCREMENT EXTENSION OF FILENAME'/
        4,10X'4 = RETURN TO PROGRAMMING MENU'/)
110     FORMAT(1X'A/I BUFFER HANDLER MENU:'//
        1,10X'1 = ERASE A/I BUFFER'/
        2,10X'2 = TYPE CONTENTS OF A/I BUFFER ON SCREEN'/
        3,10X'3 = WRITE A/I BUFFER TO FILE'/
        4,10X'4 = READ FILE INTO A/I BUFFER'/
        5,10X'5 = PLOT CONTENTS OF A/I BUFFER ON SCREEN'/
        6,10X'6 = RETURN TO PROGRAMMING MENU'//)
111     FORMAT(1X'MEASUREMENTS/CALCULATIONS SELECTION MENU:'//
        1,10X'1 = COMPUTE D = A - B BUFFERS'/
        2,10X'2 = MEASURE AMPLITUDES FROM CURSOR POSITION VALUE'/
        3,10X'3 = MEASURE AMPLITUDES FROM ACTUAL DATA VALUES'/
        4,10X'4 = INTEGRATE BETWEEN CURSOR LIMITS'/
        5,10X'5 = MOVE LAST MEASUREMENT TO NEXT A/I BUFFER ELEMENT'/
        6,10X'6 = DO QUICK-N-DIRTY MAX-MIN AMPLITUDE MEASUREMENT'/
        7,10X'7 = RETURN TO PROGRAMMING MENU'//)
112     FORMAT(1X'MISCELLANEOUS HANDLER SELECT MENU:'//
        1,10X'1 = MOVE BURNER BY M.M.'/
        2,10X'2 = MOVE BURNER BY STEPS'/
        3,10X'3 = OPEN SHUTTER'/
        4,10X'4 = CLOSE SHUTTER'/
        5,10X'5 = WAIT FOR USER TO TYPE A CARRIAGE RETURN'/
        6,10X'6 = STEP DYE LASER GRATING DRIVE MOTOR'/
        7,10X'7 = TURN ON ION COLLECTOR VOLTAGE'/
        8,10X'8 = TURN OFF ION COLLECTOR VOLTAGE'/
        9,10X'9 = RETURN TO PROGRAMMING MENU'//)
113     FORMAT(
        1'$BURNER MOVEMENT ? (+/- M.M.,DECIMAL,+ IS INTO LASER BEAM) >')
114     FORMAT
        1('$BURNER MOVEMENT ? (STEPS,INTEGER,+ IS INTO LASER BEAM) >')
115     FORMAT(E10.0)
116     FORMAT(A2)
117     FORMAT(1X'DATA ACCUMULATION SELECTION MENU:'//
        1,10X'1 = ACCUMULATE SINGLE PULSE'/
        2,10X'2 = ACCUMULATE AND AVERAGE MULTIPLE PULSES'/
        3,10X'3 = GET AND TYPE SCALE FACTORS ONLY'/
        4,10X'4 = RETURN TO PROGRAMMING MENU'//)
118     FORMAT('$LASER STEPS ? ( + IS LONGER LAMBDA, STEP = .012 NM) >')
C*******************************************************************************

        END
```

For the Liftim program, the equivalent code is:

```
      SUBROUTINE PROGRM
      INTEGER PC,SELECT
      COMMON /CNTRL/ YSCALE,XSCALE,NPULSE,MANUAL,IRFLN,ICOFLN
     1,FILE1(16),FILE2(16),FILE3(16),FILE4(16),AUTINC,NWTPLT
      COMMON /MUPROG/ INSTR(100),PC
      DATA IY/'Y'/, IA/'A'/ , IB/'B'/ , ID/'D'/
      DO 13 I=1,100
      INSTR(I)=0
13    CONTINUE
      PC = 1
      TYPE 100
1     TYPE 101
2     TYPE 102
      ACCEPT 103,SELECT
      GO TO (3,4,5,6,7,8,10,9,11)SELECT
      GO TO 1
C  ERASE BUFFERS OPCODES
3     TYPE 104
      ACCEPT 116,ICH
      IF (ICH.NE.IY)GO TO 31
      INSTR(PC)=4
      GO TO 32
31    CALL GETBUF(ICH)
      INSTR(PC)=ICH
32    PC=PC+1
      GO TO 2
      RETURN
C  ACCUMULATE SINGLE PULSE OPCODES
4     TYPE 117
45    TYPE 102
      ACCEPT 103,ICH
      GO TO (41,42,43,12)ICH
      GO TO 4
41    CALL GETBUF(ICH)
      INSTR(PC)=ICH+4
      GO TO 44
C   ACCUMULATE MULTIPLE PULSES OPCODES
42    CALL GETBUF(ICH)
      INSTR(PC)=ICH + 7
      PC=PC+1
      TYPE 106
      ACCEPT 103,INSTR(PC)
      GO TO 44
C  ACCUMUIATE SCALE FACTORS ONLY OPCODE
43    INSTR(PC)=34
44    PC=PC+1
      GO TO 45
      RETURN
C  PLOT DATA OPCODES AND ARGS
5     TYPE 107
      ACCEPT 103,ICH
      IF(ICH.LT.1.OR.ICH.GT.3)GO TO 5
      INSTR(PC)=ICH + 10
      PC = PC + 1
```

```
        DO 51 I=1,ICH
        CALL GETBUF(IVAL)
        INSTR(PC)=IVAL
        PC=PC+1
51      CONTINUE
        GO TO 12
        RETURN
C  FILE OPERATIONS SECTION
6       TYPE 108
61      TYPE 102
        ACCEPT 103,ICH
        GO TO (62,62,63,12)ICH
        GO TO 6
C  WRITE/READ DATA BUFFER OPCODES
62      CALL GETBUF(IVAL)
        IF(ICH.EQ.1)INSTR(PC)=IVAL+13
        IF(ICH.EQ.2)INSTR(PC)=IVAL+16
        PC=PC+1
        GO TO 64
C  INCREMENT EXTENSION OPCODE
63      INSTR(PC)=28
        PC=PC+1
C  GET ARG SECTION
64      CALL GETFIL(IVAL)
        INSTR(PC)=IVAL
        PC=PC+1
        GO TO 61
C  WAIT FOR CR OPCODE
76      INSTR(PC)=33
        PC=PC+1
        GO TO 12
C  AMPLITUDE/INTEGRAL BUFFER OPERATIONS
7       TYPE 110
81      TYPE 102
        ACCEPT 103,ICH
        GO TO (82,83,84,85,87,12)ICH
        GO TO 8
C  ERASE A/I OPCODE
82      INSTR(PC)=21
        GO TO 86
C  TYPE OUT CONTENTS OF A/I OPCODE
83      INSTR(PC)=26
        GO TO 86
C  WRITE A/I TO FILE OPCODE
84      INSTR(PC)=20
        GO TO 86
C  READ FILE INTO A/I OPCODE
85      INSTR(PC)=25
C  GENERAL RETURN CODE
86      PC=PC+1
        GO TO 81
C  PLOT A/I BUFFER OPCODE
87      INSTR(PC)=38
```

```
        GO TO 86
C  DATA MEASUREMENT/CALCULATIONS CODES
8       TYPE 111
91      TYPE 102
        ACCEPT 103,ICH
        GO TO (92,93,93,93,97,98,99,12)ICH
        GO TO 9
C  OPCODE FOR D=A-B OPERATION
92      INSTR(PC)=27
        GO TO 96
C  CODE TO GET BUFFER AND PLOT IT FOR AMPLITUDE MEASURE
93      CALL GETBUF(IVAL)
        INSTR(PC)=11
        PC=PC+1
        INSTR(PC)=IVAL
        PC=PC+1
        IF(ICH.EQ.3)GO TO 94
        IF(ICH.EQ.4)GO TO 95
C  OPCODE FOR AMPLITUDE FROM CURSOR POSITION
        INSTR(PC)=31
        GO TO 96
C  OPCODE FOR AMPLITUDE FROM DATA SET
94      INSTR(PC)=32
        PC=PC+1
        INSTR(PC)=IVAL
        GO TO 96
C  OPCODE FOR INTEGRATE BETWEEN LIMITS
95      INSTR(PC)=29
        PC=PC+1
        INSTR(PC)=IVAL
96      PC=PC+1
        GO TO 91
C  OPCODE FOR MOVE DATUM TO A/I BUFFER
97      INSTR(PC)=30
        GO TO 96
C  OPCODE FOR QUICK-N-DIRTY AMPLITUDE
98      INSTR(PC)=36
        PC=PC+1
        CALL GETBUF(INSTR(PC))
        GO TO 96
C  OPCODE FOR CHAIN TO FITTER ROUTINE
99      INSTR(PC) = 37
        GO TO 96
C  GO TO AUTOMATIC MODE
9       MANUAL = 0
        RETURN
10      INSTR(PC)=33
        PC=PC+1
12      IF(MANUAL.EQ.0)GO TO 1
11      RETURN

C****************** FORMAT STATEMENTS FOR SUBROUTINE PROGRM ***************
100     FORMAT(1X'******** PROGRAMMING MODE *********'//)
101     FORMAT(1X'PROGRAM SELECTION MENU:'/
```

```
        1,10X'1 = ERASE DATA BUFFER(S)'/
        2,10X'2 = DIGITIZE DATA'/
        3,10X'3 = PLOT DATA ACCUMULATED'/
        4,10X'4 = DO FILE OPERATIONS'/
        6,10X'5 = DO AMPLITUDE/INTEGRAL BUFFER OPERATIONS'/
        7,10X'6 = DO DATA MEASUREMENT/CALCULATIONS'/
        5,10X'7 = WAIT FOR USER TO TYPE A CARRIAGE RETURN'/
        8,10X'8 = GO TO AUTOMATIC MODE FROM MANUAL'/
        9,10X'9 = RETURN TO MAIN SELECT MENU'//)
102     FORMAT('$',9X'SELECTION ? >')
103     FORMAT(I5)
104     FORMAT('$ALL BUFFERS ? (Y OR N)>')
106     FORMAT('$NUMBER OF PULSES ? (1 - 32767) >')
107     FORMAT('$NUMBER OF PLOTS ON SCREEN ? (1-3) >')
108     FORMAT(1X'FILE HANDLER SELECTIONS:'//
        1,10X'1 = WRITE DATA BUFFER TO FILE'/
        2,10X'2 = READ DATA FROM FILE TO DATA BUFFER'/
        3,10X'3 = INCREMENT EXTENSION OF FILENAME'/
        4,10X'4 = RETURN TO PROGRAMMING MENU'/)
110     FORMAT(1X'A/I BUFFER HANDLER MENU:'//
        1,10X'1 = ERASE A/I BUFFER'/
        2,10X'2 = TYPE CONTENTS OF A/I BUFFER ON SCREEN'/
        3,10X'3 = WRITE A/I BUFFER TO FILE'/
        4,10X'4 = READ FILE INTO A/I BUFFER'/
        5,10X'5 = PLOT A/I BUFFER ON SCREEN'/
        6,10X'6 = RETURN TO PROGRAMMING MENU'//)
111     FORMAT(1X'MEASUREMENTS/CALCULATIONS SELECTION MENU:'//
        1,10X'1 = COMPUTE D = A - B BUFFERS'/
        2,10X'2 = MEASURE AMPLITUDES FROM CURSOR POSITION VALUE'/
        3,10X'3 = MEASURE AMPLITUDES FROM ACTUAL DATA VALUES'/
        4,10X'4 = INTEGRATE BETWEEN CURSOR LIMITS'/
        5,10X'5 = MOVE LAST MEASUREMENT TO NEXT A/I BUFFER ELEMENT'/
        6,10X'6 = DO QUICK-N-DIRTY MAX-MIN AMPLITUDE MEASUREMENT'/
        7,10X'7 = CHAIN TO FITTER PROGRAM'/
        8,10X'8 = RETURN TO PROGRAMMING MENU'//)
115     FORMAT(E10.0)
116     FORMAT(A2)
117     FORMAT(1X'DATA ACCUMULATION SELECTION MENU:'//
        1,10X'1 = ACCUMULATE SINGLE PULSE'/
        2,10X'2 = ACCUMULATE AND AVERAGE MULTIPLE PULSES'/
        3,10X'3 = GET AND TYPE SCALE FACTORS ONLY'/
        4,10X'4 = RETURN TO PROGRAMMING MENU'//)
C*************************************************************************
```

The supporting routines are the same for both programs, and the source code follows:

```
        SUBROUTINE GETBUF(ICH)
C This routine asks the user which of the 3 buffers A,B, or D is to be
C used, and returns the integers 1,2,or 3 respectively.

        DATA IA/'A'/, IB/'B'/, ID/'D'/
3       TYPE 105
        ACCEPT 116,ICH
```

```
        IF(ICH.NE.IA.AND.ICH.NE.IB.AND.ICH.NE.ID)GO TO 3
        IF(ICH.EQ.IA)ICH=1
        IF(ICH.EQ.IB)ICH=2
        IF(ICH.EQ.ID)ICH=3
105     FORMAT('$BUFFER ? (A,B,OR D) >')
116     FORMAT(A2)
        RETURN
        END

        SUBROUTINE GETFIL(ICH)
C  This routine querys the user for a file name number and returns it to
C  the calling routine
1       TYPE 109
        ACCEPT 110,ICH
        IF (ICH.EQ.1.OR.ICH.EQ.2.OR.ICH.EQ.3)RETURN
        GO TO 1
109     FORMAT('$FILE NAME NUMBER ? (1-3) >')
110     FORMAT(I3)
        END
```

The next two routines allow the user to store a microprogram in a permanent disc file, and to load the instruction array with a microprogram that was previously stored on disc.

```
        SUBROUTINE INPROG
C  Reads in microprogram to instruction array
        COMMON /MUPROG/ INSTR(100)
        LOGICAL*1 FILNAM
        DIMENSION FILNAM(16)
        CALL GETNAM(FILNAM) ! get the file name
        OPEN(UNIT=9,NAME=FILNAM,ACCESS='DIRECT',INITIALSIZE=1,
        1RECORDSIZE=128,TYPE='OLD')
        READ(9'1) (INSTR(I),I=1,100)
        CALL CLOSE(9)
        RETURN
        END

        SUBROUTINE OUTPRO
C  Writes the microprogram from the instruction array to a disc file
        COMMON /MUPROG/ INSTR(100)
        LOGICAL*1 FILNAM
        DIMENSION FILNAM(16)
        CALL GETNAM(FILNAM)
        OPEN(UNIT=9,NAME=FILNAM,ACCESS='DIRECT',INITIALSIZE=1,
        1RECORDSIZE=128,TYPE='NEW')
        WRITE(9'1) (INSTR(I),I=1,100)
        CALL CLOSE(9)
        RETURN
        END

        SUBROUTINE GETNAM(FILNAM)
C  Gets the file name for the file accesses
        LOGICAL*1 FILNAM
        DIMENSION FILNAM(16)
```

```
        TYPE 100
        ACCEPT 101, (FILNAM(I), I=1,16)
        CALL NAMCLN(FILNAM)
        RETURN
C************ FORMATS FOR ROUTINE GETNAM ***************************
100     FORMAT('$TYPE IN THE DEV:FILNAM.EXT >>>')
101     FORMAT(16A1)
C*****************************************************************
        END
        SUBROUTINE NAMCLN(FILNAM)
C  Cleans up the input file name string for use as RT-11 file name
        LOGICAL*1 DOT,SP,FILNAM(16)
        DATA DOT/'.'/, SP/' '/
C  FIND THE EXTENSION:
        I=1
2       IF(FILNAM(I).EQ.DOT)GO TO 1
        I=I+1
        GO TO 2
C  I POINTS TO THE START OF THE EXTENSION
1       I=I+1
C  CHANGE ANY SPACES IN THE EXTENSION TO ASCII ZEROS
        DO 10 J=I,I+2
        IF(FILNAM(J).EQ.SP)FILNAM(J)="60
10      CONTINUE
C  CLEAR OUT ANY JUNK AFTER THE EXTENSION, CHANGE TO ASCII NULLS
        DO 11 J=I+3,16
        FILNAM(J)=0
11      CONTINUE
        RETURN
        END
```

Both the multiphoton program and lifetime programs use the same routines, with the sole differences being in the form of the common data block. The next portion of the code deals with the execution of the microprogram. While a number of the routines are identical, there are sufficient differences due to the different graphics devices and additional hardware on the multiphoton system to make a listing of both versions desirable. First, the main execution routines. The source code for the Multiphoton program:

```
        SUBROUTINE EXECUT(NUMREM)
        INTEGER AUTINC,MANUAL,PC
        LOGICAL*1 FILE1,FILE2,FILE3,FILE4
        COMMON /CNTRL/ YSCALE,XSCALE,NPULSE,MANUAL,IRFLN,ICOFLN
        1,FILE1(16),FILE2(16),FILE3(16),FILE4(16),AUTINC
        COMMON /MUPROG/ INSTR(100)

        PC = 1
100     ICH = INSTR(PC)
        IF(ICH.NE.0)GO TO 21
        NUMREM=NUMREM-1
        TYPE 102,NUMREM
        PC=1
        IF (ICH.EQ.0.AND.NUMREM.EQ.0)RETURN
        GO TO 100
```

```
21      PC=PC+1
        GO TO (1,1,1,1,2,2,2,3,3,3,4,4,4,5,5,5,6,6,6,7,8,9
        1,10,10,11,12,13,14,15,16,17,18,19,20,22,23,25,24,24)ICH
        TYPE *,'BAD OPCODE IN ROUTINE EXECUTE!!!'
        RETURN
C  Erase buffers..
1       CALL ERABUF(ICH)
        GO TO 100
C  Accumulate single pulse from 7912AD
2       CALL ACUMSP(ICH-4)
        GO TO 100
C  Accumulate multiple pulses from 7912AD
3       CALL ACUMMP(ICH-7,INSTR(PC))
        GO TO 101
C  Make plots on graphics device of data buffers
4       IF(ICH.NE.11)GO TO 41
        CALL ERATXT
        CALL MAKPLT(1,1,1,INSTR(PC))
        GO TO 101
41      IF(ICH.NE.12)GO TO 42
        CALL ERATXT
        CALL MAKPLT(1,2,1,INSTR(PC))
        PC=PC+1
        CALL MAKPLT(1,2,2,INSTR(PC))
        GO TO 101
42      CALL ERATXT
        CALL MAKPLT(1,3,1,INSTR(PC))
        PC=PC+1
        CALL MAKPLT(1,3,2,INSTR(PC))
        PC=PC+1
        CALL MAKPLT(1,3,3,INSTR(PC))
        GO TO 101
C  Write data buffer to file
5       CALL DATOUT(ICH-13,INSTR(PC))
        GO TO 101
C  Read file into data buffer
6       CALL DATIN(ICH-16,INSTR(PC))
        GO TO 101
C  Write A/I buffer to file
7       CALL AIBOUT
        GO TO 100
C  Erase contents of A/I buffer
8       CALL ERAAIB
        GO TO 100
C  Move burner translator stepper motor
9       CALL MOVBUR(INSTR(PC))
        GO TO 101
C  Open/close shutter
10      CALL SHUTTR(ICH-23)
        GO TO 100
C  Read file into A/I buffer
11      CALL AIBIN
        GO TO 100
C  Type A/I buffer on screen of terminal
```

```
12      CALL TYPAIB
        GO TO 100
C  Put channel by channel difference of "A" and "B" into "D"
13      CALL DIFAB
        GO TO 100
C  Increment a file name
14      CALL INCEXT(INSTR(PC))
        GO TO 101
C  Do a definite integral on a data buffer
15      CALL INTGRT(INSTR(PC))
        GO TO 101
C  Move the most rescent measurement performed on the data to the
C  Next available A/I buffer location
16      CALL MV2AI
        GO TO 100
C  Measure an amplitude from the actual cursor positions
17      CALL AMPCUR
        GO TO 100
C  Measure an amplitude from the actual data values in a buffer
18      CALL AMPDAT(INSTR(PC))
        GO TO 101
C  Wait for user to type a carriage return
19      CALL WFCR
        GO TO 100
C  Acquire scale factors only from the 7912AD
20      CALL ACQSCF
        GO TO 100
C  Return code if arguments are used for opcode
101     PC=PC+1
        GO TO 100
C  Step the laser wavelength
22      CALL LASER(INSTR(PC))
        GO TO 101
C  Do a quick - and - dirty amplitude measurement
23      CALL AMPQND(INSTR(PC))
        GO TO 101
C  Turn on/off high voltage switch
24      CALL JOLTS(ICH-37)
        GO TO 100
C  Plot contents of A/I buffer on graphics device
25      CALL PLTAIB
        GO TO 100
102     FORMAT(1X,'NUMBER OF EXECUTIONS REMAINING = ',I4)
        END
```

With the addition of the chaining opcode, and the deletion of the additional hardware support, the same routine for the lifetime program is:

```
        SUBROUTINE EXECUT(NUMREM,CHNIN)
        INTEGER AUTINC,MANUAL,PC,CHNIN
        LOGICAL*1 FILE1,FILE2,FILE3,FILE4
        COMMON /CNTRL/ YSCALE,XSCALE,NPULSE,MANUAL,IRFLN,ICOFLN
        1,FILE1(16),FILE2(16),FILE3(16),FILE4(16),AUTINC,NWTPLT
        COMMON /MUPROG/ INSTR(100),PC
```

```
C  Detect chain-in, and continue with old pc if so...
        IF (CHNIN.EQ.0)GO TO 98
        CHNIN=0
        GO TO 100
98      PC = 1
100     ICH = INSTR(PC)
        IF(ICH.NE.0)GO TO 21
        NUMREM=NUMREM-1
        TYPE 102,NUMREM
        PC=1
        IF (ICH.EQ.0.AND.NUMREM.EQ.0)GO TO 120
        GO TO 100
21      PC=PC+1
        GO TO (1,1,1,1,2,2,2,3,3,3,4,4,4,5,5,5,6,6,6,7,8,9
        1,10,10,11,12,13,14,15,16,17,18,19,20,22,23,24,25)ICH
        TYPE *,'BAD OPCODE IN ROUTINE EXECUTE!!!'
        GO TO 120
1       CALL ERABUF(ICH)
        GO TO 100
2       CALL ACUMSP(ICH-4)
        GO TO 100
3       CALL ACUMMP(ICH-7,INSTR(PC))
        GO TO 101
4       IF(ICH.NE.11)GO TO 41
        CALL MAKPLT(1,1,INSTR(PC))
        IF(INSTR(PC+1).NE.29.AND.INSTR(PC+1).NE.31
        1.AND.INSTR(PC+1).NE.32)GO TO 43
        GO TO 101
41      IF(ICH.NE.12)GO TO 42
        CALL MAKPLT(2,1,INSTR(PC))
        PC=PC+1
        CALL MAKPLT(2,2,INSTR(PC))
        GO TO 43
42      CALL MAKPLT(3,1,INSTR(PC))
        PC=PC+1
        CALL MAKPLT(3,2,INSTR(PC))
        PC=PC+1
        CALL MAKPLT(3,3,INSTR(PC))
43      IF(NWTPLT.NE.1)PAUSE 'TYPE RETURN TO CONTINUE'
        CALL ERASE
        CALL VTMODE
        CALL VTPAGE
        GO TO 101
5       CALL DATOUT(ICH-13,INSTR(PC))
        GO TO 101
6       CALL DATIN(ICH-16,INSTR(PC))
        GO TO 101
7       CALL AIBOUT
        GO TO 100
8       CALL ERAAIB
        GO TO 100
9       GO TO 100
10      GO TO 100
11      CALL AIBIN
```

```
        GO TO 100
12      CALL TYPAIB
        GO TO 100
13      CALL DIFAB
        GO TO 100
14      CALL INCEXT(INSTR(PC))
        GO TO 101
15      CALL INTGRT(INSTR(PC))
        GO TO 101
16      CALL MV2AI
        GO TO 100
17      CALL AMPCUR
        GO TO 100
18      CALL AMPDAT(INSTR(PC))
        GO TO 101
19      CALL WFCR
        GO TO 100
20      CALL ACQSCF
        GO TO 100
101     PC=PC+1
        GO TO 100
22      GO TO 100
23      CALL AMPQND(INSTR(PC))
        GO TO 101
C  This is the call to chain to another program..
24      CALL CHNFIT(NUMREM)
        GO TO 120
25      CALL PLTAIB
        GO TO 100
102     FORMAT(1X,'NUMBER OF EXECUTIONS REMAINING = ',I4)
120     RETURN
        END
```

The routines that are common to both programs differ only in the form of their common blocks in the data structures, and will not be listed twice. These routines are the true "work horses" of the programs, and actually perform the useful functions. These are for the multiphoton program, but are more or less identical to the ones for the lifetime program.

```
        SUBROUTINE WFCR
C  This routine prompts the user to type a carriage return, and waits
C  for it. This is useful to allow user interaction with the experiment
C  before continuing to the next microprogram instruction.

        TYPE 100
        ACCEPT 101,DUMMY
        RETURN
100     FORMAT(1X,'WAITING FOR CARRIAGE RETURN...')
101     FORMAT(A4)
        END

        SUBROUTINE GOTOMA
C  This routine clears the instruction array and sets the system to
C  manual mode. It is called by routine EXECUT
```

```
      COMMON /CNTRL/ YSCALE,XSCALE,NPULSE,MANUAL,IRFLN,ICOFLN
      1,FILE1(16),FILE2(16),FILE3(16),FILE4(16),AUTINC
      COMMON /MUPROG/ INSTR(100)

      DO 10 I=1,100
      INSTR(I)=0
10    CONTINUE
      MANUAL = 1
      RETURN
      END

      SUBROUTINE ERABUF(IBUF)
C  This routine erases data buffers A,B, and D. If IBUF=1, only "A" is
C  erased. If IBUF=2, then only "B" is erased. If IBUF=3, then only "D"
C  is erased. If IBUF=4, then all 3 are erased.

      COMMON /BUFFA/ RADATA(512)
      COMMON /BUFFB/ RBDATA(512)
      COMMON /BUFFD/ RDDATA(512)

      DO 10 I=1,512
      IF(IBUF.EQ.1.OR.IBUF.EQ.4)RADATA(I)=0.0
      IF(IBUF.EQ.2.OR.IBUF.EQ.4)RBDATA(I)=0.0
      IF(IBUF.EQ.3.OR.IBUF.EQ.4)RDDATA(I)=0.0
10    CONTINUE
      RETURN
      END
```

The following routines comprise the actual data taking package:

```
      SUBROUTINE ACQSCF
C  This routine is the controlling routine to acquire the horizontal and
C  vertical scale factors from the 7912AD digitizer. It reads in the
C  scale factors, and then converts the ASCII to internal floating point
C  numbers by calling CVTSCF. The results are printed, and stored in
C  common. It is called by routines EXECUT, ACUMMP, and ACUMSP
      EXTERNAL GETSCF
      COMMON /CNTRL/ YSCALE,XSCALE,NPULSE,MANUAL,IRFLN,ICOFLN
      1,FILE1(16),FILE2(16),FILE3(16),FILE4(16),AUTINC

      CALL CIF(GETSCF)          ! execute the PDB "GETSCF"
      CALL CVTSCF               !convert ASCII to internal format
      TYPE 106,YSCALE,XSCALE    ! Type the scale factors to user
      RETURN
C********************* FORMATS ************************************
106   FORMAT(1X'VERTICAL SCALE = ',E8.2,' V/DIV',/
      1,1X'HORIZONTAL SCALE = ',E8.2,' T/DIV'/)
C*********************************************************************

      END

      SUBROUTINE CVTSCF
C  This routine converts the ASCII encoded scale factors sent to the
C  scale factor string arrays by the 7912AD into floating point
```

```
C  numerical form. It calls routine SORTIT, and is called by ACQSCF
        LOGICAL*1 YSCALS,HSCALS
        COMMON /VSCALF/ YSCALS(20)
        COMMON /HSCALF/ HSCALS(20)
        COMMON /CNTRL/ YSCALE,XSCALE,NPULSE,MANUAL,IRFLN,ICOFLN
        1,FILE1(16),FILE2(16),FILE3(16),FILE4(16),AUTINC

        CALL SORTIT(YSCALS,YSCALE)
        CALL SORTIT(HSCALS,XSCALE)
        RETURN
        END

        SUBROUTINE SORTIT(BUFF1,VALUE)
        LOGICAL*1 BUFF1(20),BUFF2(15),SP,E,TEMP(6)
        DATA SP/' '/,E/'E'/
C  This routine is the workhorse of routines ACQSCF and CVTSCF.....

C   FIRST FIND THE SPACE AFTER THE V/D OR H/D...
        I=1
1       IF(BUFF1(I).EQ.SP)GO TO 2
        I=I+1
        GO TO 1
C   AND POINT AT THE FIRST CHAR OF THE NUMBER
2       I=I+2
C   THEN UN-SWITCH ALL THE NUMBERS, AND PUT IN BUFF2
        DO 3 J=1,9,2
        BUFF2(J)=BUFF1(I+1)
        BUFF2(J+1)=BUFF1(I)
        I=I+2
3       CONTINUE
C   AND THEN FIND THE 'E' IN THE EXPRESSION
        DO 4 I=1,10
        IF(BUFF2(I).EQ.E)GO TO 5
4       CONTINUE
C   AND PUT A NULL BYTE AFTER THE 3 SUBSEQUENT CHARACTERS
5       BUFF2(I+4)=0
        J=I-1
        DECODE(J,100,BUFF2)ACCUM
        J=I+1
        DECODE(3,101,BUFF2(J))IPWR
        VALUE=ACCUM * (10. ** IPWR)
        RETURN
C************************ FORMAT STATEMENTS ******************************
100     FORMAT(F10.3)
101     FORMAT(I3)
C*************************************************************************
        END

        SUBROUTINE ACUMSP(IBUF)
C  This is the driver routine for acquiring single digitized pulses from
C  the 7912AD. It calls routine CIF and CVTSCF, and is called by EXECUT.
        INTEGER DATA
        COMMON /DATA/ DATA(512)
        COMMON /BUFFA/ RADATA(512)
```

```
        COMMON /BUFFB/ RBDATA(512)
        COMMON /BUFFD/ RDDATA(512)
        EXTERNAL DUMP,GETSCF
        CALL CIF(DUMP)  ! execute the PDB to get and dump 1 pulse
        DO 5 I=1,512
        IF(IBUF.EQ.1)RADATA(I)=DATA(I)/2.       ! do this since data is sent
        IF(IBUF.EQ.2)RBDATA(I)=DATA(I)/2.       ! multiplied by 2
        IF(IBUF.EQ.3)RDDATA(I)=DATA(I)/2.
5       CONTINUE
        CALL CIF(GETSCF)        ! get scale factors
        CALL CVTSCF             ! convert scale factors
        RETURN
        END

        SUBROUTINE ACUMMP(IBUF,NPULSE)
C  This is the driver routine for acquiring multiple signal-averaged
C  pulses from the 7912AD digitizer. The digitize-signal average mode
C  of the 7912 is used, with the numbers sent as the required powers
C  of two.

        INTEGER DATA
        LOGICAL*1 NUMSA
        COMMON /DATA/ DATA(512)
        COMMON /NUMSA/ NUMSA(4)
        EXTERNAL DUMPSA,DUMP,GETSCF

        IPULSE = NPULSE
6       IF(IPULSE.LT.64)GO TO 8         ! if doing less than 64, go to 8
C ***** NUMSA=64 *******                 !else, digitize and signal average
        ENCODE(2,104,NUMSA)64           !64 scans, decrement the total number
        CALL CIF(DUMPSA)                !to be done, and sum into the data
        IPULSE=IPULSE-64                !array, normalizing for the total
        CALL SUMIT(IBUF,NPULSE)         !number to be done.
        GO TO 6
8       IF(IPULSE.LT.32)GO TO 9         !if doing < 32, go to 9
C****** NUMSA=32 *******                 !else, repeat the above operations for
        ENCODE(2,104,NUMSA)32           !32 pulses
        CALL CIF(DUMPSA)
        IPULSE=IPULSE-32
        CALL SUMIT(IBUF,NPULSE)
9       IF(IPULSE.LT.16)GO TO 10
C****** NUMSA=16 ********                ! same thing but for 16 pulses
        ENCODE(2,104,NUMSA)16
        CALL CIF(DUMPSA)
        IPULSE=IPULSE-16
        CALL SUMIT(IBUF,NPULSE)
10      IF(IPULSE.LT.8)GO TO 11
C****** NUMSA=8 *********            ! same thing, but for 8 pulses
        ENCODE(2,104,NUMSA)08
        CALL CIF(DUMPSA)
        IPULSE=IPULSE-8
        CALL SUMIT(IBUF,NPULSE)
11      DO 12 I=1,IPULSE            ! for less than 8, do the rest one-by-one
        CALL CIF(DUMP)
```

```
        DO 5 J = 1,512
        DATA(J) = DATA(J)/2. + 0.5
5       CONTINUE
        CALL SUMIT(IBUF,NPULSE)
12      CONTINUE
        CALL CIF(GETSCF)                !and acquire scale factors.
        CALL CVTSCF
        RETURN
104     FORMAT(I2)
        END

        SUBROUTINE SUMIT(IBUF,NPULSE)
C  This routine sums in the data sent from the 7912AD digitizer
C  contained in the array "DATA" into the data buffer indicated by the
C  user. The data is normalized to the number of scans to be done in
C  total for this accumulation.  NOTE:  Old data is not automatically
C  erased, but rather is summed into.  Erasure of old data MUST be done
C  via user instruction.  Also, normalization of data to differing scale
C  factors is not done.

        INTEGER DATA
        COMMON /DATA/ DATA(512)
        COMMON /BUFFA/ RADATA(512)
        COMMON /BUFFB/ RBDATA(512)
        COMMON /BUFFD/ RDDATA(512)

        DO 10 I=1,512
        IF(IBUF.EQ.1)RADATA(I)=RADATA(I) + DATA(I)/FLOAT(NPULSE)
        IF(IBUF.EQ.2)RBDATA(I)=RBDATA(I) + DATA(I)/FLOAT(NPULSE)
        IF(IBUF.EQ.3)RDDATA(I)=RDDATA(I) + DATA(I)/FLOAT(NPULSE)
10      CONTINUE
        RETURN
        END
```

Next, the data file handling routines, and their associated routines:

```
        SUBROUTINE INCEXT(IFILE)
C  This routine finds and increments by 1 the NUMERICAL extension of the
C  file name stored in the string specified by IFILE. IFILE is the file
C  name number. IFILE=1 is FILE1, IFILE=2 is FILE2, and IFILE=3 is FILE3.
C  It calls routine INCR to do the actual work. It is called by EXECUT

        INTEGER AUTINC
        LOGICAL*1 FILE1,FILE2,FILE3
        COMMON /CNTRL/ YSCALE,XSCALE,NPULSE,MANUAL,IRFLN,ICOFLN
        1,FILE1(16),FILE2(16),FILE3(16),FILE4(16),AUTINC
        IF(IFILE.EQ.1)CALL INCR(FILE1)
        RETURN
        IF(IFILE.EQ.2)CALL INCR(FILE2)
        RETURN
        IF(IFILE.EQ.3)CALL INCR(FILE3)
        RETURN
        END
```

```
      SUBROUTINE INCR(FILNAM)
C  This is the workhorse routine that increments file name extensions.
C  It does so by finding the period between the filename and extension,
C  decoding the ASCII to an integer, incrementing the integer, and then
C  encoding back to ASCII and replacing the original extension in the
C  string.

      LOGICAL*1 FILNAM
      DIMENSION FILNAM(16)
      LOGICAL*1 DOT,SP
      DATA DOT/'.'/, SP/' '/
C    FIND THE FILENAME EXTENSION:
      I=1
2     IF(FILNAM(I).EQ.DOT)GO TO 1
      I=I+1
      GO TO 2
C  WHEN THE EXT IS FOUND, PUT THE POINTER ON THE FIRST CHARACTER OF IT.
1     I=I+1
C    NOW CONVERT THE 3 ASCII CHARACTERS TO AN INTEGER:
      DECODE(3,103,FILNAM(I))INDEX
C    THEN INCREMENT THAT INTEGER:
      INDEX=INDEX+1
C    THEN CONVERT THE INTEGER BACK TO THE 3 ASCII CHARACTER EQUIVALENT:
      ENCODE(3,103,FILNAM(I))INDEX
C    BUT LEADING ZEROS ARE CONVERTED TO ASCII SPACE CHARACTERS, SO
C  FIND ANY IN THE EXTENSION, AND CHANGE TO ASCII ZEROS:
      DO 10 J=I,I+2
      IF(FILNAM(J).EQ.SP)FILNAM(J)="60
10    CONTINUE
C    FOLLOWING THE EXTENSION, THE OPEN STAT ENT WANTS NULL BYTES, SO
C    GIVE EM TO HIM:
      DO 11 J=I+3,16
      FILNAM(J)=0
11    CONTINUE
      RETURN
C************************** FORMAT STATEMENTS ******************************
103   FORMAT(I3)
C***************************************************************************
      END


      SUBROUTINE DATIN(IBUF,IFILE)
C  This is the driver routine for reading data files on disc into the
C  data buffers in the program. It calls routine INP. IBUF = 1,2,or 3
C  for buffers A,B, or D respectively as the data destination. IFILE
C  = 1,2, or 3 indicating that the routine should use the string stored
C  in FILE1, FILE2, or FILE3 respectively as the file name.

      INTEGER AUTINC
      LOGICAL*1 FILE1,FILE2,FILE3,FILE4
      COMMON /BUFFA/ RADATA(512)
      COMMON /BUFFB/ RBDATA(512)
      COMMON /BUFFD/ RDDATA(512)
      COMMON /CNTRL/ YSCALE,XSCALE,NPULSE,MANUAL,IRFLN,ICOFLN
```

```
     1,FILE1(16),FILE2(16),FILE3(16),FILE4(16),AUTINC
      IF(IBUF.NE.1)GO TO 1
      IF(IFILE.EQ.1)CALL INP(RADATA,FILE1)
      IF(IFILE.EQ.2)CALL INP(RADATA,FILE2)
      IF(IFILE.EQ.3)CALL INP(RADATA,FILE3)
      GO TO 10
1     IF(IBUF.NE.2)GO TO 2
      IF(IFILE.EQ.1)CALL INP(RBDATA,FILE1)
      IF(IFILE.EQ.2)CALL INP(RBDATA,FILE2)
      IF(IFILE.EQ.3)CALL INP(RBDATA,FILE3)
      GO TO 10
2     IF(IFILE.EQ.1)CALL INP(RDDATA,FILE1)
      IF(IFILE.EQ.2)CALL INP(RDDATA,FILE2)
      IF(IFILE.EQ.3)CALL INP(RDDATA,FILE3)
10    RETURN
      END

      SUBROUTINE INP(RBUF,FILNAM)
C  This is the workhorse routine that reads in disc files of data. RBUF
C  is the actual array name of the buffer to receive the data, and FILNAM
C  is the actual ASCII string that contains the file name. Note that if
C  the request file names flag (IRFLN) is set, the FILNAM is requested,
C  and will be used as the source of the data. Also, that file name will
C  replace the one sent in the argument of the call.

      LOGICAL*1 FILNAM,FILE1,FILE2,FILE3,FILE4,IDINF1,IDINF2
      INTEGER AUTINC
      DIMENSION RBUF(512),FILNAM(16)
      COMMON /CNTRL/ YSCALE,XSCALE,NPULSE,MANUAL,IRFLN,ICOFLN
     1,FILE1(16),FILE2(16),FILE3(16),FILE4(16),AUTINC
      COMMON /ID/ IDINF1(72),IDINF2(72)

      IF(IRFLN.EQ.1)CALL GETNAM(FILNAM)
      OPEN(UNIT=9,NAME=FILNAM,ACCESS='DIRECT',INITIALSIZE=5,
     1RECORDSIZE=640,TYPE='OLD')
      READ(9'1) (RBUF(I),I=1,512),YSCALE,XSCALE,(IDINF1(I),I=1,72)
     1,(IDINF2(I),I=1,72)
      CALL CLOSE(9)
C  Echo file name if flag is set...
      IF(ICOFLN.EQ.1)TYPE 100,(FILNAM(I),I=1,16)
C  Increment the filename extension if the autoincrement flag is set...
      IF(AUTINC.NE.0)CALL INCR(FILNAM)
      RETURN
100   FORMAT(1X'FILE READ : ',16A1)
      END

      SUBROUTINE DATOUT(IBUF,IFILE)
C  This is the driver routine for the output of the data buffers to disc
C  data files. It corresponds closely to the DATIN routine.
C  The arguments are as defined in DATIN.

      INTEGER AUTINC
      LOGICAL*1 FILE1,FILE2,FILE3,FILE4
      COMMON /BUFFA/ RADATA(512)
```

```
      COMMON /BUFFB/ RBDATA(512)
      COMMON /BUFFD/ RDDATA(512)
      COMMON /CNTRL/ YSCALE,XSCALE,NPULSE,MANUAL,IRFLN,ICOFLN
     1,FILE1(16),FILE2(16),FILE3(16),FILE4(16),AUTINC

      IF(IBUF.NE.1)GO TO 1
      IF(IFILE.EQ.1)CALL OUP(RADATA,FILE1)
      IF(IFILE.EQ.2)CALL OUP(RADATA,FILE2)
      IF(IFILE.EQ.3)CALL OUP(RADATA,FILE3)
      GO TO 10
1     IF(IBUF.NE.2)GO TO 2
      IF(IFILE.EQ.1)CALL OUP(RBDATA,FILE1)
      IF(IFILE.EQ.2)CALL OUP(RBDATA,FILE2)
      IF(IFILE.EQ.3)CALL OUP(RBDATA,FILE3)
      GO TO 10
2     IF(IFILE.EQ.1)CALL OUP(RDDATA,FILE1)
      IF(IFILE.EQ.2)CALL OUP(RDDATA,FILE2)
      IF(IFILE.EQ.3)CALL OUP(RDDATA,FILE3)
10    RETURN
      END

      SUBROUTINE OUP(RBUF,FILNAM)
C  This is the workhorse routine for output of data buffers to disc data
C  files. See routine INP for description.

      LOGICAL*1 FILNAM,FILE1,FILE2,FILE3,FILE4,IDINF1,IDINF2
      INTEGER AUTINC
      DIMENSION RBUF(512),FILNAM(16)
      COMMON /CNTRL/ YSCALE,XSCALE,NPULSE,MANUAL,IRFLN,ICOFLN
     1,FILE1(16),FILE2(16),FILE3(16),FILE4(16),AUTINC
      COMMON /ID/ IDINF1(72),IDINF2(72)

      IF(IRFLN.EQ.1)CALL GETNAM(FILNAM)
      OPEN(UNIT=9,NAME=FILNAM,ACCESS='DIRECT',INITIALSIZE=5,
     1RECORDSIZE=640,TYPE='NEW')
      WRITE(9'1) (RBUF(I),I=1,512),YSCALE,XSCALE,(IDINF1(I),I=1,72)
     1,(IDINF2(I),I=1,72)
      CALL CLOSE(9)
      IF(ICOFLN.EQ.1)TYPE 100,(FILNAM(I),I=1,16)
      IF(AUTINC.NE.0)CALL INCR(FILNAM)
      RETURN
100   FORMAT(1X'FILE WRITTEN : ',16A1)
      END

      SUBROUTINE TYPAIB
C  This routine types the current contents of the A/I buffer on the
C  screen of the user terminal, in 5 columns, 20 rows.
      COMMON /AI/ AIX,AIY,AIVALS(100),IPTR
      DO 10 I=1,100,5
      TYPE 102, (AIVALS(J),J=I,I+4)
10    CONTINUE
      RETURN
102   FORMAT(1X,5E14.5)
103   FORMAT(1X'AMPLITUDE/INTEGRAL INFORMATION BUFFER CONTENTS:')
```

```
      END
      SUBROUTINE ERAAIB
C  This routine erases the contents of the A/I buffer, and initializes
C  the A/I buffer pointer to 1
      COMMON /AI/ AIX,AIY,AIVALS(100),IPTR
      DO 10 I=1,100
      AIVALS(I)=0.0
10    CONTINUE
      IPTR = 1
      RETURN
      END

      SUBROUTINE MV2AI
C  This routine moves the latest measurement being held in variable
C  "AIY" into the A/I buffer position indicated by "IPTR", and updates
C  IPTR to point to the next free space in the A/I buffer array.
      COMMON /AI/ AIX,AIY,AIVALS(100),IPTR
      AIVALS(IPTR)=AIY
      IPTR=IPTR+1
      RETURN
      END

      SUBROUTINE AIBIN
C  This routine reads a disc file into the contents of the A/I buffer
C  and sets IPTR to point at the first available element.
      INTEGER DATA,AUTINC,SELECT,MANUAL
      LOGICAL*1 FILE1,FILE2,FILE3,YSCALS,HSCALS,NUMSA
      LOGICAL*1 IDINF1,IDINF2,FILE4
      COMMON /CNTRL/ YSCALE,XSCALE,NPULSE,MANUAL,IRFLN,ICOFLN
      1,FILE1(16),FILE2(16),FILE3(16),FILE4(16),AUTINC
      COMMON /ID/ IDINF1(72),IDINF2(72)
      COMMON /AI/ AIX,AIY,AIVALS(100),IPTR

      IF(IRFLN.EQ.1)CALL GETNAM(FILE4)  !get file name if flag is set
C  open the old file for reading into buffer
      OPEN(UNIT=9,NAME=FILE4,ACCESS='DIRECT',INITIALSIZE=2,
      1RECORDSIZE=256,TYPE='OLD')
C  read it in...
      READ(9'1) (AIVALS(I),I=1,100),(IDINF1(I),I=1,72)
      1,(IDINF2(I),I=1,72)
C  close the channel, since we're done..
      CALL CLOSE(9)
      IPTR = 1
C  set the pointer to the first available location above the data
      DO 10 I = 100,1,-1
      IF(AIVALS(I).EQ.0)GO TO 10
      IPTR = I
      GO TO 12
10    CONTINUE
```

```
C  Echo the file name if flag is set..
11      IF(ICOFLN.EQ.1)TYPE 100,(FILE4(I),I=1,16)
C  Increment the file name if the flag is set...
        IF(AUTINC.EQ.1)CALL INCR(FILE4)
        RETURN
100     FORMAT(1X'FILE READ : ',16A1)
        END


        SUBROUTINE AIBOUT
C  This routine writes the current contents of the A/I buffer to a disc
C  file. The contents of the buffer and pointer are unchanged.
        INTEGER DATA,AUTINC,SELECT,MANUAL
        LOGICAL*1 FILE1,FILE2,FILE3,YSCALS,HSCALS,NUMSA
        LOGICAL*1 IDINF1,IDINF2,FILE4
        COMMON /CNTRL/ YSCALE,XSCALE,NPULSE,MANUAL,IRFLN,ICOFLN
        1,FILE1(16),FILE2(16),FILE3(16),FILE4(16),AUTINC
        COMMON /ID/ IDINF1(72),IDINF2(72)
        COMMON /AI/ AIX,AIY,AIVALS(100),IPTR

C  Request file name if flag is set:
        IF(IRFLN.EQ.1)CALL GETNAM(FILE4)
C  Open the file for output from buffer to file:
        OPEN(UNIT=9,NAME=FILE4,ACCESS='DIRECT',INITIALSIZE=2,
        1RECORDSIZE=256,TYPE='NEW')
C  Write the buffer to the file..
        WRITE(9'1) (AIVALS(I),I=1,100),(IDINF1(I),I=1,72)
        1,(IDINF2(I),I=1,72)
C  Close the file
        CALL CLOSE(9)
C  Echo the file name if the flag is set
        IF(ICOFLN.EQ.1)TYPE 100,(FILE4(I),I=1,16)
C  Increment the file name if the flag is set
        IF(AUTINC.EQ.1)CALL INCR(FILE4)
        RETURN
100     FORMAT(1X'FILE WRITTEN : ',16A1)
        END

        SUBROUTINE AMPQND(IBUFF)
C  This routine does a "Quick-and-Dirty" amplitude measurement on the
C  data in a buffer. It does so by calling a routine to find the maximum
C  in the data, and then calculates a baseline from the average of the
C  first 15 data points in the buffer. The amplitude is then the
C  difference of these two numbers.

        COMMON /BUFFA/ RADATA(512)
        COMMON /BUFFB/ RBDATA(512)
        COMMON /BUFFD/ RDDATA(512)
        COMMON /PLT/ SCALF,YMIN,YMAX
        COMMON /AI/ AIX,AIY,AIVALS(100),IPTR

        GO TO (1,2,3)IBUFF
        RETURN
C  Find max, min for buffer "A", and the baseline
```

```
1       CALL SCANER(RADATA)
        CALL BASLIN(RADATA,BASE)
        GO TO 4
C  Find max, min for buffer "B", and the baseline
2       CALL SCANER(RBDATA)
        CALL BASLIN(RBDATA,BASE)
        GO TO 4
C  Find max, min for buffer "D", and the baseline
3       CALL SCANER(RDDATA)
        CALL BASLIN(RDDATA,BASE)
C  Compute amplitude, put in AIY in common.
4       AIY = YMAX-BASE
        AIX = 0
        TYPE *,'AMPLITUDE = ',AIY
        RETURN
        END

C*********************************************************************

        SUBROUTINE BASLIN(RBUFF,BASE)
C  CALCULATES THE AVERAGE VALUE OF THE FIRST 15 DATA IN A BUFFER FOR USE AS
C  A BASELINE FOR QUICK-AND-DIRTY AMPLITUDE MEASUREMENTS
        DIMENSION RBUFF(512)
        BASE = 0.
        DO 10 I = 1,15
        BASE = BASE + RBUFF(I)
10      CONTINUE
        BASE = BASE/15.
        RETURN
        END


        SUBROUTINE AMPDAT(IBUFF)
C  This is the driver routine for amplitudes from data values.
C  This routine measures amplitudes from the data set. First, a user
C  cursor position is queried for and received. The "X" value of this
C  is used to get an index into a data buffer array, and get a data
C  value. The process is repeated, and the amplitude determined to be
C  the difference of the two data values.

        COMMON /BUFFA/ RADATA(512)
        COMMON /BUFFB/ RBDATA(512)
        COMMON /BUFFD/ RDDATA(512)
        IF (IBUFF.EQ.1)CALL AMPLIT(1,RADATA)
        IF (IBUFF.EQ.2)CALL AMPLIT(1,RBDATA)
        IF (IBUFF.EQ.3)CALL AMPLIT(1,RDDATA)
        RETURN
        END

        SUBROUTINE AMPCUR
C  This routine is the driver for amplitudes from actual cursor values.
C  This is desirable when looking at noisy data where amplitudes can be
C  "guesstimated" by moving the cursor to the best user estimated points
C  and differencing the two values to get an amplitude.
```

```
        CALL AMPLIT(0,IBUFF)
        RETURN
        END



        SUBROUTINE AMPLIT(MODE,BUFFER)
C  This routine is the workhorse of the amplitude measuring routines. As
C  it is dependent on the graphics device in use, the versions are
C  different. This one is for the Multiphoton program.

        DIMENSION BUFFER(512)
        COMMON /AI/ AIX,AIY,AIVALS(100),IPTR

C  Get a cursor value, and return either the cursor data or data set
C  data, depending on the value of "MODE".
        CALL GETVAL(MODE,AIX1,AIY1,BUFFER)
        CALL GETVAL(MODE,AIX2,AIY2,BUFFER)
        AIX=AIX1
        AIY=ABS(AIY1-AIY2)
        TYPE *,'X-VALUE = ',AIX,' Y-VALUE = ',AIY
        RETURN
        END
```

The routine "AMPLIT" for the liftime program is

```
        SUBROUTINE AMPLIT(MODE,BUFFER)
        DIMENSION BUFFER(512)
        COMMON /AI/ AIX,AIY,AIVALS(100),IPTR

        CALL GETVAL(MODE,AIX1,AIY1,BUFFER)
        CALL GETVAL(MODE+2,AIX2,AIY2,BUFFER)
        AIX=AIX1
        AIY=ABS(AIY1-AIY2)
        CALL ERASE      !clear graphics terminal screen
        CALL VTMODE     !set to VT-100 mode
        CALL VTPAGE     !clear VT-100 screen
        TYPE *,'X-VALUE = ',AIX,' Y-VALUE = ',AIY
        RETURN
        END


        SUBROUTINE DIFAB
C  This routine differences Buffer "A" and "B" and places the
C  difference in buffer "D". This method is used for noise subtraction.

        COMMON /BUFFA/ RADATA(512)
        COMMON /BUFFB/ RBDATA(512)
        COMMON /BUFFD/ RDDATA(512)

        DO 10 I=1,512
        RDDATA(I)=RADATA(I)-RBDATA(I)
```

```
10      CONTINUE
        RETURN
        END

        SUBROUTINE INTGRT(IBUFF)
C  This routine is the driver for performing definite integrals on data
C  sets stored in the buffers. First, the data is displayed on the
C  graphics device, then the user moves a cursor to the starting X,Y
C  value for the integral and accepts that value. Next, the user moves
C  the cursor to the ending X,Y value and accepts that value. The routine
C  then computes the value of the integral using the trapezoidal rule,
C  prints the value, and stores it in common. If desired, a later
C  instruction can move it to the A/I buffer.
C  An important feature to note is that this
C  integration was done for the purpose of finding the
C  absolute areas. Any areas that fall "below" the Baseline
C  are Added, not subtracted from the area integrated
C  above the Baseline. This was done because the integral
C  Represented a number of ions, not a mathematical
C  4-quadrant function.

        COMMON /BUFFA/ RADATA(512)
        COMMON /BUFFB/ RBDATA(512)
        COMMON /BUFFD/ RDDATA(512)
        COMMON /AI/ AIX,AIY,AIVALS(100),IPTR

        CALL GETVAL(2,X1,Y1,RADATA)     ! get the starting X,Y
        CALL GETVAL(3,X2,Y2,RADATA)     ! get the ending X,Y
C  Do the trapezoidal rule integration..
        IF (IBUFF.EO.1) CALL TRAP(RADATA,X1,Y1,X2,Y2,AINT)
        IF (IBUFF.EQ.2) CALL TRAP(RBDATA,X1,Y1,X2,Y2,AINT)
        IF (IBUFF.EO.3) CALL TRAP(RDDATA,X1,Y1,X2,Y2,AINT)
        AIY = AINT      !store the result in common
        TYPE *,'THE VALUE OF THE INTEGRAL IS:',AINT
        RETURN
        END
```

The same routine for LIFTIM, which differs due to graphics hardware differences is:

```
        SUBROUTINE INTGRT(IBUFF)
        COMMON /BUFFA/ RADATA(512)
        COMMON /BUFFB/ RBDATA(512)
        COMMON /BUFFD/ RDDATA(512)
        COMMON /AI/ AIX,AIY,AIVALS(100),IPTR

        CALL GETVAL(0,X1,Y1,RADATA)
        CALL GETVAL(3,X2,Y2,RADATA)
        CALL ERASE      ! erase screen
        CALL VTMODE     ! set to VT-100 mode
        CALL VTPAGE     ! clear VT-100 screen
        IF (IBUFF.EO.1) CALL TRAP(RADATA,X1,Y1,X2,Y2,AINT)
        IF (IBUFF.EO.2) CALL TRAP(RBDATA,X1,Y1,X2,Y2,AINT)
        IF (IBUFF.EO.3) CALL TRAP(RDDATA,X1,Y1,X2,Y2,AINT)
        AIY = AINT
```

```
      TYPE *,'THE VALUE OF THE INTEGRAL IS:',AINT
      RETURN
      END

      SUBROUTINE TRAP(RARRY,X1,Y1,X2,Y2,VAL)
C  This is the trapezoidal rule integration routine proper...
C  RARRY is the data buffer containing the data set upon which the
C  integration is to be done. X1,Y1 forms the starting channel and
C  baseline endpoint, X2,Y2 forms the ending channel and baseline point.
C  VAL contains the value of the completed integral.
      DIMENSION RARRY(512)
      RIY=Y1
      RJY=Y2
      IX=X1
      JX=X2
      DELTAY=RJY-RIY
      DELTAX=JX-IX
      SLOPE=DELTAY/DELTAX
      INCX=ABS(DELTAX)/DELTAX
      YINC=SLOPE*INCX
      VAL=0.
      KX=IX
      BASEY=RIY
1     VAL=VAL+0.5*(RARRY(KX)-BASEY+RARRY(KX+INCX)-(BASEY+YINC))
      BASEY=BASEY+YINC
      KX=KX+INCX
      IF(KX.NE.JX)GO TO 1
      RETURN
      END
```

For the Multiphoton program, routine GETVAL is:

```
      SUBROUTINE GETVAL(MODE,XVAL,YVAL,BUFFER)
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C  MODE = 0 IS AMPLITUDE FROM CURSOR                        C
C  MODE = 1 IS AMPLITUDE FROM DATA BUFFER                   C
C  MODE = 2 IS FIRST INTEGRAL LIMIT FROM CURSOR             C
C  MODE = 3 IS SECOND INTEGRAL LIMIT FROM CURSOR            C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      COMMON /PLT/ SCALF,YMIN,YMAX
      COMMON /CURSOR/ IDX1,RDY1,IDX2,RDY2,IWAIT,IDFSE1,IDFSE2
      COMMON /PLTLIM/ IFIRST,ILAST
      DIMENSION BUFFER(512)
      BYTE A,L,R,U,D,ICHAR,S,ONE,TWO
      DATA A/'A'/,R/'R'/,L/'L'/,U/'U'/,D/'D'/,S/'S'/
      DATA ONE /'1'/, TWO/'2'/
C   Wait for MDP to settle...
      CALL WFMDP(M1,M2,M3,M4)
C   Initialize cursors
      CALL INITCU
C   Calculate a center screen starting point
      ICHAN = (ILAST - IFIRST)/2.
```

```
        YVAL  = (YMAX-YMIN)/2 + YMIN
        IF (MODE.EQ.1)YVAL = BUFFER(ICHAN)
C   Take care of default cursor positions, if any
        IF(MODE.EQ.3)GO TO 3
        IF (IDFSE1.NE.1)GO TO 4
        ICHAN  = IDX1
        YVAL = RDY1
        GO TO 4
3       IF(IDFSE2.NE.1)GO TO 4
        ICHAN = IDX2
        YVAL = RDY2
C   Set terminal handler to special mode.. No character echo, characters
C   available immediately to user program with no carriage return
C   needed.
4       I = IPEEK("44)
        J = I + "10000
        CALL IPOKE("44,J)
C   Put some labels on the top of the screen
        IF(MODE.EQ.0)CALL LAB1
        IF(MODE.EQ.1)CALL LAB2
        IF(MODE.EQ.2.OR.MODE.EQ.3)CALL LAB4
C   Set the cursor movement step size at 1 pixel
        ISTEP = 1
C   Calculate the screen (512 X 512) Y coordinate from the Y data value
10      IY = ((YVAL-YMIN)*SCALF + 220)/8.
C   Calculate the screen X coordinate from the X data value
        IX = ((ICHAN-IFIRST)*3553./(ILAST-IFIRST) + 364)/8.
C   Erase old cursors
        CALL ERACUR
C   draw new cursors
        CALL DRWCUR(IX,IY)
C   Type out cursor position on MDP
        CALL LAB3(ICHAN,YVAL)
C   If not going to wait for user, go to 50
        IF(IWAIT.EQ.0)GO TO 50
C   Else, get a character from the user
15      K=ITTINR()
        IF(K.LT.0)GO TO 15
        ICHAR = K
C   User wants to accept cursor value, go to 50
        IF(ICHAR.EQ.A)GO TO 50
C   User wants to move cursor up, go to 11
        IF(ICHAR.EQ.U)GO TO 11
C   User wants to move cursor down, go to 12
        IF(ICHAR.EQ.D)GO TO 12
C   User wants to move cursor left, go to 14
        IF(ICHAR.EQ.L)GO TO 14
C   User wants to move cursor right, go to 13
        IF(ICHAR.EQ.R)GO TO 13
C   User wants to reset movement step size, go to 16
        IF(ICHAR.EQ.S)GO TO 16
C   User wants to accept default position 1
        IF(ICHAR.EQ.ONE)GO TO 17
C   User wants to accept default position 2
```

```
        IF(ICHAR.EQ.TWO)GO TO 18
        GO TO 15
C   Code to move cursor up
11      IF(MODE.EQ.1)GO TO 15
        IF ((YVAL+ISTEP+.5).GE.YMAX)GO TO 15
        YVAL=YVAL+ISTEP
        GO TO 10
C   Code to move cursor down
12      IF(MODE.EQ.1)GO TO 15
        IF((YVAL-ISTEP).LE.YMIN)GO TO 15
        YVAL = YVAL-ISTEP
        GO TO 10
C   Code to move cursor right
13      IF((ICHAN+ISTEP).GT.511)GO TO 15
        ICHAN = ICHAN + ISTEP
        IF(MODE.EQ.1)YVAL=BUFFER(ICHAN)
        GO TO 10
C   Code to move cursor left
14      IF((ICHAN-ISTEP).LT.0)GO TO 15
        ICHAN = ICHAN -ISTEP
        IF(MODE.EQ.1)YVAL=BUFFER(ICHAN)
        GO TO 10
C   Code to get new step size from user
16      CALL IPOKE("44,I)  !restore to normal terminal handler mode
        TYPE 100
        ACCEPT *,ISTEP
100     FORMAT('$STEP SIZE? >')
        CALL IPOKE("44,J)  !back to special terminal mode
        GO TO 15
C   Code to load default cursor position 1 from current position
17      IDX1 = ICHAN
        RDY1 = YVAL
        IDFSE1 = 1
        GO TO 15
C   Code to load default cursor position 2 from current position
18      IDX2 = ICHAN
        RDY2 = YVAL
        IDFSE2 = 1
        GO TO 15
C   Common exit code...
50      CALL IPOKE("44,I)
        XVAL = ICHAN
        RETURN
        END
```

Since the lifetime program has a different graphics output device, routine GETVAL for that program looks as follows:

```
        SUBROUTINE GETVAL(MODE,XVAL,YVAL,BUFFER)
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C  MODE = 0 IS XVAL,YVAL FROM CURSOR                C
C  MODE = 1 IS YVAL FROM DATA BUFFER,ELEMENT XVAL   C
C  MODE = 2 IS MODE 0 FOR SECOND POINT,CURSOR       C
C  MODE = 3 IS MODE 1 FOR SECOND POINT, BUFFER      C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
        COMMON /PLT/ SCALF,YMIN,YMAX
        COMMON /PLTLIM/ IFIRST,ILAST
        DIMENSION BUFFER(512)
        BYTE IANS
        INTEGER X

        YSCALE = 2946. / (YMAX - YMIN)
        J = 3100
10      CONTINUE
C  Get a set of coordinates from the cursors
        CALL GRAFIN(IX,IY)
C  Convert them to 4096 X 4096 space
        IX=IX*4
        IY=IY*4
C  put the cursor out of the way for printing on screen
        IF (MODE.EQ.1.OR.MODE.EQ.0)CALL MOVE(0,J)
        IF (MODE.EQ.2.OR.MODE.EQ.3)CALL MOVE(2048,J)
        CALL ALFMOD
C  Find what the data values are for the screen coordinates
        X = IFIX((IX - 364.)*(ILAST-IFIRST) / 3563. + IFIRST + 0.5)
        IF (MODE.EQ.0.OR.MODE.EQ.2)Y = (IY - 220) / YSCALE + YMIN
        IF (MODE.EQ.1.OR.MODE.EQ.3)Y = BUFFER(X)
C ask for user acceptance.
        TYPE 101,X,Y
        J = J - 88
        ACCEPT 100,IANS
        IF (IANS.NE.'Y')GO TO 10
C store the accepted values
        XVAL = X
        YVAL = Y
        RETURN
100     FORMAT(A1)
101     FORMAT('$X = ',I4,'  Y = ',F6.2,'  ACCEPT? >')
        END
```

The next sections of code are the graphics routines. The first of these, SCANER, is common to both programs, and follows:

```
        SUBROUTINE SCANER(RDATA,ISIZE)
C  SCANS DATA SUBRANGE AND FINDS THE MIN AND MAX IN BUFFER
        COMMON /PLT/ SCALF,DYMIN,DYMAX
        COMMON /PLTLIM/ IFIRST,ILAST
        DIMENSION RDATA(ISIZE)
        DYMIN = 1E38
        DYMAX = -1E38
        DO 10 I=IFIRST,ILAST
        IF(RDATA(I).LT.DYMIN)DYMIN=RDATA(I)
```

```
10      CONTINUE
        RETURN
        END
```

The next sections are the graphics package for the multiphoton program, and support the MDP-3A 512 X 512 raster scan graphics processor.

```
        SUBROUTINE SETMDP
        DIMENSION IARRY(50)
        BYTE RED(32)
        DATA RED/"0,"21,"42,"63,"104,"125,"146,"167,"210,"231
        1, "252,"273,"314,"335,"356,"377,"0,"21,"42,"63,"104
        2, "125,"146,"167,"210,"231,"252,"273,"314,"335,"356,"377/

        CALL INITGA(IARRY,50)           !initializes GOS arrays
        CALL GOSIN(IARRY)               !initializes MDP unit
        CALL CLRMDP                     !clears all registers of MDP
        CALL DMA(IARRY,"102420,16,RED)  !loads color lookup tables
        CALL SETMOD(IARRY,512)          !sets MDP to 512 X 512 mode
        CALL SETCLR(IARRY,15)           !sets default color to white
        CALL DCLEAR(IARRY)              !clears MDP data memory
        CALL SETGR(IARRY)               !turns on graphics display
        CALL STARTG(IARRY)              !starts execution of above..
        CALL WFMDP(MD1,MD2,MD3,MD4)     !waits until MDP is done
        RETURN
        END

        SUBROUTINE MAKPLT(MODE,NUMPLT,POSPLT,NUMBUF)
        INTEGER POSPLT

C  Set screen window low Y coordinate to 0
        MINSY = 0
C  If first plot of 2 or 3, or only plot to be done, initialize for plots
        IF(NUMPLT.EQ.1.OR.POSPLT.EQ.1)CALL SETMDP
        IF (NUMPLT.NE.1)GO TO 1
C  Set screen window high Y coordinate to 3210
        MAXSY = 3210
C  put 1 picture (plot) on screen, from buffer number "NUMBUF", in the
C  screen window defined by MINSY,MAXSY
        CALL PICTUR(MODE,NUMBUF,MINSY,MAXSY)
        RETURN
1       IF (NUMPLT.NE. 2)GO TO 2
C  come here for two plots on the screen at once
        IF(POSPLT.NE.1)GO TO 3
C  posplt = 1 puts plot on lower half of screen
        MAXSY = 1604
        CALL PICTUR(MODE,NUMBUF,MINSY,MAXSY)
        RETURN
C  posplt = 2 puts plot on upper half of screen
3       MINSY = 1605
        MAXSY = 3210
        CALL PICTUR(MODE,NUMBUF,MINSY,MAXSY)
        RETURN
2       GO TO (4,5,6)POSPLT
```

```
4       MAXSY = 1069
        CALL PICTUR(MODE,NUMBUF,MINSY,MAXSY)
        RETURN
C  plot in middle third of screen
5       MINSY = 1070
        MAXSY = 2139
        CALL PICTUR(MODE,NUMBUF,MINSY,MAXSY)
        RETURN
C  plot in top third of screen
6       MINSY = 2140
        MAXSY = 3210
        CALL PICTUR(MODE,NUMBUF,MINSY,MAXSY)
        RETURN
        END


        SUBROUTINE PICTUR(MODE,IBUFF,MINSY,MAXSY)
C  this routine makes the plot in the screen window indicated in
C  MINSY, MAXSY. MODE is used to indicate how to label the picture.
        COMMON /BUFFA/ RADATA(512)
        COMMON /BUFFB/ RBDATA(512)
        COMMON /BUFFD/ RDDATA(512)
C  draw the axis in the screen window
        CALL AXES(MINSY,MAXSY)
        GO TO (1,2,3)IBUFF
C  find the Max and MIN of buffer "A"
1       CALL SCANER(RADATA)
C  plot the data in buffer "A" in the window
        CALL PLOTER(RADATA,MINSY,MAXSY)
        GO TO 4
C  find the Max and MIN of buffer "B"
2       CALL SCANER(RBDATA)
C  plot the data in buffer "B" in the window
        CALL PLOTER(RBDATA,MINSY,MAXSY)
        GO TO 4
C  find the Max and MIN of buffer "D"
3       CALL SCANER(RDDATA)
C  plot the data in buffer "D" in the window
        CALL PLOTER(RDDATA,MINSY,MAXSY)
C  put the labels on the axes, and the plot heading
4       CALL LABMDP(MODE,MINSY,MAXSY)
        RETURN
        END

        SUBROUTINE AXES(MINSY,MAXSY)
C  This routine draws the axes in the screen window.
        INTEGER DELTAX,DELTAY,Y1,X1
        DELTAX = 713
C  Define X1, Y1: the coordinates of the axes origin
        X1 = 364
        Y1 = MINSY + 220
C  draw the "Y" axis
        CALL DRWMDP(X1,(MAXSY - 44),X1,Y1)
C  draw the "X" axis
```

```
C  put the tick marks on the "Y" axis
        DELTAY = (MAXSY - MINSY - 264) / 5.0
        Y1 = MINSY + 220 + DELTAY
        DO 10 I=1,5
        CALL DRWMDP(336,Y1,392,Y1)
        Y1 = Y1 + DELTAY
10      CONTINUE
C  put the tick marks on the "X" axis
        X1 = 364 + DELTAX
        DO 15 I = 1,5
        CALL DRWMDP(X1,(MINSY + 176),X1,(MINSY + 264))
        X1 = X1 + DELTAX
15      CONTINUE
        RETURN
        END


        SUBROUTINE PLOTER(RDATA,MINSY,MAXSY,ISIZE)
C  PLOTS DATA SUBRANGE ON SCREEN WINDOW SELECTED
        COMMON /PLT/ YSCALE,DYMIN,DYMAX
        COMMON /PLTLIM/ IFIRST,ILAST
        DIMENSION RDATA(ISIZE)
        INTEGER LOX,LOY,YLENGT,Y1,X1

        LOX = 364     !SCREEN WINDOW LOW X VALUE FOR PLOTTING
        HIX = 3927    !SCREEN WINDOW HIGH X VALUE
        XLENGT = 3563  !SCREEN WINDOW LENGTH
        LOY = MINSY + 220 !SCREEN WINDOW LOWEST Y VALUE
        LOLDX = LOX
        LOLDY = LOY
        YLENGT = MAXSY - MINSY - 264  ! PLOT WINDOW HEIGHT
        XSCALE = FLOAT(XLENGT) / FLOAT(ILAST-IFIRST) !CALCULATE DATA X EXTENT
        YSCALE = FLOAT(YLENGT) / FLOAT(DYMAX - DYMIN)
C  NOW PLOT THE DATA IN THE PLOTTING WINDOW GIVEN
        DO 10 I = IFIRST,ILAST
        X1 = (I-IFIRST) * XSCALE + LOX + 0.5
        Y1 = (RDATA(I) - DYMIN) * YSCALE + LOY + 0.5
        CALL DRWMDP(LOLDX,LOLDY,X1,Y1)
        LOLDX = X1
        LOLDY = Y1
10      CONTINUE
        RETURN
        END

        SUBROUTINE LABMDP(MODE,MINSY,MAXSY)
C  PUTS LABELS ON THE AXES FOR THE DATA SET AND SUBRANGE
        COMMON /PLT/ SCALF,DYMIN,DYMAX
        COMMON /PLTLIM/ IFIRST,ILAST
        BYTE NAMSTR(6)
        INTEGER XDATA , XCHANG, DELTAX, DELTAY, Y1, I, X1
C   FIRST INDEX OF YLIN IS GRAPH POSITION, SECOND IS GRAPH NUMBER
C   1 PLOT = (I,1) 2 PLOTS ,BOTTOM = (I,2) TOP= (I,3)
C   3 PLOTS BOTTOM=(I,4) MIDDLE=(I,5) TOP=(I,6)
        INTEGER YLIN(6,6),XLIN(5)
```

```
     1,17,15,13,11,9,7,30,29,28,26,25,24,21,20,19,17,16,15
     2,13,12,11,9,8,7/

     DATA XLIN/7,13,18,24,29/
     BYTE LAB1(24),LAB2(20),LAB3(20),LAB4(22),LAB5(22)
C    Label 1 is: PLOT OF DIGITIZED DATA
     DATA LAB1 /'P','L','O','T',' ','O','F',' ','D','I','G','I'
     1, 'T','I','Z','E','D',' ','D','A','T','A',0,0/
C    Label 3 is: SETUP RUN DISPLAYED
     DATA LAB3 /'S','E','T','U','P',' ','R','U','N',' ','D','I'
     2, 'S','P','L','A','Y','E','D',0/
C    Label 2 is: DATA RUN DISPLAYED
     DATA LAB2 /'D','A','T','A',' ','R','U','N',' ','D','I'
     2, 'S','P','L','A','Y','E','D',0,0/
C    Label 4 is: PLOT OF MEASUREMENT
     DATA LAB4 /'P','L','O','T',' ','O','F',' ','M','E','A','S'
     2,'U','R','E','M','E','N','T','S',0,0/
C    Label 5 is: A/I BUFFER DISPLAYED
     DATA LAB5 /'A','/','I',' ','B','U','F','F','E','R',' ','D'
     2,'I','S','P','L','A','Y','E','D',0,0/

C    Set up proper screen position plotting position indices:
     IF(MINSY.EQ.0 .AND. MAXSY.EQ.3210)IPLT=1
     IF(MINSY.EQ.0 .AND. MAXSY.EQ.1604)IPLT=2
     IF(MINSY.EQ.1605 .AND. MAXSY.EQ.3210)IPLT=3
     IF(MINSY.EQ.0 .AND. MAXSY.EQ.1069)IPLT=4
     IF(MINSY.EQ.1070 .AND. MAXSY.EQ.2139)IPLT=5
     IF(MINSY.EQ.2140 .AND. MAXSY.EQ.3210)IPLT=6
C    Set up some constants for computing label values
     DELTAX = 713
     DELTAY = 589
     Y1 = 220
     YCHANG = (DYMAX - DYMIN) / 5.0
     YDATA = DYMIN
C    Put labels on ticks of Y axis, computing correct values
     DO 10 I = 1,6
     IYDATA=YDATA
     ENCODE(3,100,NAMSTR(1))IYDATA
     NAMSTR(4)=0
     NAMSTR(5)=0
     CALL TYPMDP(0,YLIN(I,IPLT),3,NAMSTR)
     YDATA = DYMIN + YCHANG * (I-1)
10   CONTINUE
C    Put labels on ticks of X axis, computing correct values
     XCHANG = FLOAT(ILAST-IFIRST)/5.0  ! X-CHANGE TO EACH TICK
     DO 15 I = 1,5
     XDATA = IFIX(I * XCHANG +0.5 + IFIRST)
     ENCODE(3,100,NAMSTR(1))XDATA
     NAMSTR(4)=0
     NAMSTR(5)=0
     CALL TYPMDP(XLIN(I),31,3,NAMSTR)
15   CONTINUE
C    And now, put on the correct picture headings...
     IF(MODE.EQ.1.OR.MODE.EQ.2)CALL TYPMDP(0,0,12,LAB1)
```

```
      IF(MODE.EQ.0)CALL TYPMDP(0,1,10,LAB3)
      IF (MODE.EQ.1.OR.MODE.EQ.2)RETURN
      IF(MODE.EQ.2)CALL TYPMDP(0,0,11,LAB4)
      IF(MODE.EQ.2)CALL TYPMDP(0,0,11,LAB5)
100   FORMAT(I3)
      RETURN
      END

      SUBROUTINE DRWMDP(ISX,ISY,IX,IY)
C  This routine draws a line on the MDP screen between ISX,ISY and
C  IX,IY.  The data is assumed to be scaled to 4096 X 4096 space.
      DIMENSION IARRY(140)
C  rescale the data to 512 X 512 space
      IJX=ISX*512./4096.
      IJY=ISY*512./4096.
      JJX=IX*512./4096.
      JJY=IY*512./4096.
      CALL INITGA(IARRY,140)                  !initialize GOS array
      CALL DLINE(IARRY,IJX,IJY,JJX,JJY)       !draw line in MDP
      CALL STARTG(IARRY)                      !start execution of GOS list
      CALL WFMDP(MD1,MD2,MD3,MD4)             !wait until MSDP is done.
      RETURN
      END

      SUBROUTINE ERATXT
C  This routine erases all text in the MDP text overlay
      BYTE FF(4)
      DATA FF/"14,0,0,0/
      DIMENSION IARRY(30)
      CALL INITGA(IARRY,30)
      CALL SETTXT(IARRY)
      CALL TEXT(IARRY,0,0,FF)
      CALL STARTG(IARRY)
      CALL WFMDP(MD1,MD2,MD3,MD4)
      RETURN
      END

      SUBROUTINE TYPMDP(IXL,IYL,ISIZ,ICHAR)
C  This routine puts text on the screen of the MDP. The text is
C  contained in the array ICHAR of dimension ISIZ, and the text is
C  to be placed on the screen in line number IYL from the top, and
C  starting IXL characters from the left of the screen.
      DIMENSION ICHAR(ISIZ)
      DIMENSION IARRY(30)
      CALL INITGA(IARRY,30)
      CALL SETTXT(IARRY)
      CALL TEXT(IARRY,IXL,IYL,ICHAR(1))
      CALL STARTG(IARRY)
      CALL WFMDP(MD1,MD2,MD3,MD4)
      RETURN
      END
```

```
      SUBROUTINE LAB1
C  This routine puts the label "AMPLITUDE FROM CURSOR POSITION" on the
C  MDP screen.
      DIMENSION ICHAR(32)
      BYTE ICHAR
      DATA ICHAR /'A','M','P','L','I','T','U','D','E',' '
     1,'F','R','O','M',' ','C','U','R','S','O','R',' ','P'
     2,'O','S','I','T','I','O','N',0,0/
      CALL TYPMDP(0,0,32,ICHAR)
      RETURN
      END


      SUBROUTINE LAB2
C  This routine puts the label "AMPLITUDE FROM DATA SET" on the MDP
C  screen.
      DIMENSION ICHAR(24)
      BYTE ICHAR
      DATA ICHAR /'A','M','P','L','I','T','U','D','E',' '
     1,'F','R','O','M',' ','D','A','T','A',' ','S','E','T',0/
      CALL TYPMDP(0,0,32,ICHAR)
      RETURN
      END


      SUBROUTINE LAB3(IX,YVAL)
C  This routine prints out the channel number and amplitude contained in
C  IX and YVAL respectively on the MDP screen.
      DIMENSION ICHAR(20),JCHAR(16),KCHAR(8)
      BYTE ICHAR,JCHAR,KCHAR
      DATA ICHAR /'C','H','A','N','N','E','L',' ','N'
     1,'U','M','B','E','R',' ',' ','=',' ',' ',' ',' ',0/
      DATA JCHAR /'A','M','P','L','I','T','U','D','E',' '
     1,'=',' ',' ',' ',' ',0,0/
      CALL TYPMDP(0,1,18,ICHAR)
      CALL TYPMDP(0,2,14,JCHAR)
      ENCODE(3,100,KCHAR)IX
      KCHAR(4) = 0
      CALL TYPMDP(18,1,4,KCHAR)
      ENCODE(7,101,KCHAR)YVAL
      KCHAR(8) = 0
      CALL TYPMDP(13,2,8,KCHAR)
      RETURN
100   FORMAT(I3)
101   FORMAT(F7.2)
      END


      SUBROUTINE LAB4
C  This routine puts the label "INTEGRAL FROM DATA SET" on the MDP
C  screen.
      DIMENSION ICHAR(24)
      BYTE ICHAR
      DATA ICHAR /'I','N','T','E','G','R','A','L',' '
```

```
      CALL TYPMDP(0,0,24,ICHAR)
      RETURN
      END
      SUBROUTINE INITCU
C  This routine turns on overlay memory #0 display, clears it, and
C  prepares for the display of cursors.
      DIMENSION IARRY(30)
      CALL INITGA(IARRY,30)    !initialize GOS array
      CALL SETOVR(IARRY,0)     !set default overlay to 0
      CALL SETOVD(IARRY,64)    !turn on display of overlay 0
      CALL OCLEAR(IARRY)       !clear default overlay
      CALL STARTG(IARRY)       !execute instructions in GOS list
      CALL WFMDP(M1,M2,M3,M4) !wait until MDP is done.
      RETURN
      END

      SUBROUTINE ERACUR
C  This routine erases the cursors in the overlay memory.
      DIMENSION IARRY(25)
      CALL INITGA(IARRY,25)
      CALL OCLEAR(IARRY)
      CALL STARTG(IARRY)
      CALL WFMDP(M1,M2,M3,M4)
      RETURN
      END

      SUBROUTINE DRWCUR(IXL,IYL)
C  This routine draws the cursors in overlay memory
      DIMENSION IARRY(50)
      INTEGER X,Y,Y1,Y2
      CALL INITGA(IARRY,50)
      X = IXL/4
      Y = IYL/2
      Y1 = Y-1
      Y2 = Y+1
20    IF(Y1.GE.0)GO TO 25
      Y1 = Y1 + 1
      GO TO 20
25    IF(Y2.LE.255)GO TO 30
      Y2=Y2-1
      GO TO 25
30    CALL OPOINT(IARRY,1,X,Y)
      CALL OLINE(IARRY,1,X,Y1,X,Y2)
      CALL STARTG(IARRY)
      CALL WFMDP(M1,M2,M3,M4)
      RETURN
      END


      SUBROUTINE PLTAIB
C  MAIN DRIVER FOR PLOTTING A/I BUFFER ON MDP GRAPHICS SCREENS
      COMMON /AI/ AIX,AIY,AIVALS(100),IPTR
      COMMON /PLT/ SCALF,DATMIN,DATMAX
      COMMON /PLTLIM/ IFIRST,ILAST
```

```
C  SET UP TO USE SAME PLOT ROUTINES AS FOR DATA FILES
        ISTOR1 = IFIRST
        ISTOR2 = ILAST
        IFIRST = 1
        ILAST = IPTR-1
        MINSY = 0
        MAXSY = 3210
C  INITIALIZE FOR PLOTTING
        CALL SETMDP
C  DRAW THE AXES TO PLOT ON
        CALL AXES(MINSY,MAXSY)
C  SCAN THE DATA FOR MIN, MAX
        CALL SCANER(AIVALS,100)
C  PLOT THE DATA IN THE BUFFER
        CALL PLOTER(AIVALS,MINSY,MAXSY,100)
C  PUT ON THE LABELS...
        CALL LABMDP(2,MINSY,MAXSY)
        IFIRST = ISTOR1
        ILAST = ISTOR2
        RETURN
        END
```

The graphics package for the liftim program is:

```
        SUBROUTINE MAKPLT(NUMPLT,POSPLT,NUMBUF)
        INTEGER POSPLT
C  Set screen window low Y coordinate to 0
        MINSY = 0
C  if only 1 plot or first of several on same screen, initialize for
C  plotting.
        IF(NUMPLT.EQ.1.OR.POSPLT.EQ.1)CALL INITT
        IF (NUMPLT.NE.1)GO TO 1
C  if only one plot, set screen max Y=3210, and make picture...
        MAXSY = 3210
        CALL PICTUR(NUMBUF,MINSY,MAXSY)
        RETURN
1       IF (NUMPLT.NE. 2)GO TO 2
C  Come here if going to make 2 pictures on same screen
        IF(POSPLT.NE.1)GO TO 3
C  Make bottom one...
        MAXSY = 1604
        CALL PICTUR(NUMBUF,MINSY,MAXSY)
        RETURN
C  make top one...
3       MINSY = 1605
        MAXSY = 3210
        CALL PICTUR(NUMBUF,MINSY,MAXSY)
        RETURN
C  Come here to put three plots on same screen
2       GO TO (4,5,6)POSPLT
C  make bottom one..
4       MAXSY = 1069
        CALL PICTUR(NUMBUF,MINSY,MAXSY)
        RETURN
```

```
5       MINSY = 1070
        MAXSY = 2139
        CALL PICTUR(NUMBUF,MINSY,MAXSY)
        RETURN
C  make top one...
6       MINSY = 2140
        MAXSY = 3210
        CALL PICTUR(NUMBUF,MINSY,MAXSY)
        RETURN
        END


        SUBROUTINE PICTUR(IBUFF,MINSY,MAXSY)
C  Routine to make a complete plot in the given screen window.
        COMMON /BUFFA/ RADATA(512)
        COMMON /BUFFB/ RBDATA(512)
        COMMON /BUFFD/ RDDATA(512)
C  Draw axes and tick marks:
        CALL AXES(MINSY,MAXSY)
        GO TO (1,2,3)IBUFF
C  Get Max and Min in buffer "A"
1       CALL SCANER(RADATA)
C  Plot data in buffer "A"
        CALL PLOTER(RADATA,MINSY,MAXSY)
        GO TO 4
C  Get Max and Min in buffer "B"
2       CALL SCANER(RBDATA)
C  Plot data in buffer "B"
        CALL PLOTER(RBDATA,MINSY,MAXSY)
        GO TO 4
C  Get Max and Min in buffer "D"
3       CALL SCANER(RDDATA)
C  Plot data in buffer "D"
        CALL PLOTER(RDDATA,MINSY,MAXSY)
C  Put labels on axes.
4       CALL LABEL(MINSY,MAXSY)
        RETURN
        END



        SUBROUTINE AXES(SCRYMN,SCRYMX)
C  This routine is responsible for putting the labels on the X and Y axes
C  of the plots.
        INTEGER DELTAY,Y1,I,X1,SCRYMN,SCRYMX
        DELTAX = 713
C  X1,Y1 are the coordinates of the origin of the axes
        X1 = 364
        Y1 = SCRYMN + 220
C  move to the top of the Y axis
        CALL MOVE(X1,(SCRYMX - 44))
C  and draw to the origin
        CALL DRAW(X1,Y1)
C  and draw to the end of the X axis
```

```
        CALL DRAW(3927,Y1)
C  put on the Y-axis ticks
        DELTAY = (SCRYMX - SCRYMN - 264) / 5
        Y1 = SCRYMN + 220 + DELTAY
        DO 10 I = 1,5
        CALL MOVE(336,Y1)
        CALL DRAW(392,Y1)
        Y1 = Y1 + DELTAY
10      CONTINUE
C  put on the X-axis ticks
        X1 = 364 + DELTAX
        DO 20 I = 1,5
        CALL MOVE(X1,(SCRYMN + 176))
        CALL DRAW(X1,(SCRYMN + 264))
        X1 = X1 + DELTAX
20      CONTINUE
        RETURN
        END



        SUBROUTINE PLOTER(BUFF,SCRYMN,SCRYMX)
C  This routine plots the data subrange of buffer "BUFF" in the screen window
C  defined by SCRYMN, SCRYMX the Y-value Minimum and Maximum
C  respectively.

        DIMENSION BUFF(512)
        COMMON /PLTLIM/ IFIRST,ILAST
        COMMON /PLT/ YSCALE,DATYMN,DATYMX
        INTEGER LOY,YLENGT,Y1,I,X1,SCRYMN,SCRYMX
        REAL Y,XSCALE,YSCALE

        LOX = 364     !SCREEN WINDOW LOW X VALUE FOR PLOTTING
        HIX = 3927    !SCREEN WINDOW HIGH X VALUE
        XLENGT = 3563  !SCREEN WINDOW LENGTH
        LOY = SCRYMN + 220    !SCREEN WINDOW LOW Y FOR PLOTTING
        CALL MOVE(LOX,LOY)  !GO TO FIRST PLOTTING POINT
        YLENGT = SCRYMX - SCRYMN - 264    !CALCULATE PLOT WINDOW HEIGHT
        XSCALE = XLENGT / (ILAST-IFIRST) !CALCULATE DATA X EXTENT
        YSCALE = YLENGT / (DATYMX - DATYMN)  !SAME FOR DATA Y
C  NOW PLOT THE DATA IN THE PLOTTING WINDOW GIVEN
        DO 10  I = IFIRST,ILAST
        X1 = (I-IFIRST) * XSCALE + LOX
        Y1 = (BUFF(I) - DATYMN) * YSCALE + LOY
        J = 0
        IF ((Y1 .LE. SCRYMX) .AND. (Y1 .GE. SCRYMN))J = 1
        IF(J.EQ.0)CALL MOVE(X1,Y1)
        IF(J.EQ.1)CALL DRAW(X1,Y1)
10      CONTINUE
        CALL ALFMOD
        RETURN
        END
```

```
      SUBROUTINE LABEL(SCRYMN,SCRYMX)
C  PUTS LABELS ON THE AXES FOR THE DATA SET AND SUBRANGE

      COMMON /PLT/ SCALF,DATYMN,DATYMX
      COMMON /PLTLIM/ IFIRST,ILAST
      LOGICAL*1 NAMSTR(10)
      INTEGER XDATA,DELTAY,Y1,X1,SCRYMN,SCRYMX
C  SET UP FOR LABELLING Y-AXIS
      DELTAY = (SCRYMX - SCRYMN - 264) / 5  !SAME THING FOR Y
      Y1 = SCRYMN + 220  ! Y1 IS LOWEST POINT ON Y-AXIS
      YCHANG = (DATYMX - DATYMN) / 5.0  ! Y-INCREMENT TO NEXT TICK
C  PUT ON Y-AXIS LABELS .....
      DO 10 I = 1,6
      CALL MOVE(0,Y1)
      CALL ALFMOD
      IYDATA =IFIX((I-1)*YCHANG + DATYMN +0.5)
      ENCODE(6,100,NAMSTR)IYDATA
      NAMSTR(7)=0
      CALL LINOUT(NAMSTR)
      Y1 = Y1 + DELTAY
10    CONTINUE
C  SET UP FOR LABELLING X-AXIS...
      DELTAX = 713   ! LENGTH OF THE USABLE X-AXIS BETWEEN TICKS
      X1 = 364 + DELTAX  ! LOWEST POINT ON X-AXIS,SCREEN COORDINATE
      XCHANG = FLOAT(ILAST-IFIRST)/5.0  ! X-CHANGE TO EACH TICK
      DO 20 I = 1,5
      CALL MOVE((X1 - 84),(SCRYMN + 44))
      XDATA = IFIX(I * XCHANG +0.5 + IFIRST)
      CALL ALFMOD
      ENCODE(4,101,NAMSTR)XDATA
      NAMSTR(5)=0
      CALL LINOUT(NAMSTR)
      X1 = X1 + DELTAX
20    CONTINUE
      RETURN
C******************** FORMAT STATEMENTS ******************************
100   FORMAT(I6)
101   FORMAT(I4)
C*********************************************************************
      END


      SUBROUTINE PLTAIB
C  MAIN DRIVER FOR PLOTTING A/I BUFFER ON VT-100 OR TEKTRONIX 4014 SCREENS
      INTEGER AUTINC,MANUAL
      LOGICAL*1 FILE1,FILE2,FILE3,FILE4
      COMMON /CNTRL/ YSCALE,XSCALE,NPULSE,MANUAL,IRFLN,ICOFLN
      1,FILE1(16),FILE2(16),FILE3(16),FILE4(16),AUTINC,NWTPLT
      COMMON /AI/ AIX,AIY,AIVALS(100),IPTR
      COMMON /PLT/ SCALF,DATMIN,DATMAX
      COMMON /PLTLIM/ IFIRST,ILAST
```

```
C  SET UP TO USE SAME PLOT ROUTINES
        ISTOR1 = IFIRST
        ISTOR2 = ILAST
        IFIRST = 1
        ILAST = IPTR-1
        MINSY = 0
        MAXSY = 3210
C  INITIALIZE FOR PLOTTING
        CALL INITT
C  DRAW THE AXES TO PLOT ON
        CALL AXES(MINSY,MAXSY)
C  SCAN THE DATA FOR MIN, MAX
        DATMIN = 1.0E38
        DATMAX = -1.0E38
        DO 10 I = 1,IPTR-1
        IF(AIVALS(I).GT.DATMAX)DATMAX=AIVALS(I)
        IF(AIVALS(I).LT.DATMIN)DATMIN=AIVALS(I)
10      CONTINUE
C  PLOT THE DATA IN THE A/I BUFFER
        CALL MOVE(364,220)  !GO TO FIRST PLOTTING POINT
        XSCALE = 3563. / (IPTR-1) !CALCULATE DATA X EXTENT
        YSCALE = 2946. / (DATMAX - DATMIN)  !SAME FOR DATA Y
C  NOW PLOT THE DATA IN THE PLOTTING WINDOW GIVEN
        DO 20  I = 1,IPTR
        IX1 = (I-1) * XSCALE + 364
        IY1 = (AIVALS(I) - DATMIN) * YSCALE + 220
        CALL DRAW(IX1,IY1)
20      CONTINUE
        CALL ALFMOD
        CALL LABEL(MINSY,MAXSY)
        IFIRST = ISTOR1
        ILAST = ISTOR2
        IF(NWTPLT.NE.1)PAUSE 'TYPE RETURN TO CONTINUE'
        CALL ERASE
        CALL VTMODE
        CALL VTPAGE
        RETURN
        END



        SUBROUTINE INITT
C  This routine initializes the terminal for plotting.
        CALL SETTRM             !initializes plotting library software
        CALL VTPAGE             !clears VT-100 screen
        CALL TKMODE             !sets VT-100 to Tektronix look alike mode
        CALL ERASE              !erases Tektronix screen
        DO 10 I = 1,32767       !short settling delay
10      CONTINUE
        CALL BELL               !beep terminal bell
        RETURN
        END
```

The next segments of code are the program listing routines. The function of these routines in the programs is to provide a convenient means to see, in plain English, what sequence of microprogramming instructions has been entered. This is also useful as a means to verify the contents of a microprogram that was read in from disc. Both versions of the routine are shown, as there is sufficient difference to warrant the additional space required. First, the version for the multiphoton program:

```
      SUBROUTINE LISTPR
      INTEGER AUTINC,PC
      LOGICAL*1 FILE1,FILE2,FILE3,YSCALS,HSCALS,NUMSA
      LOGICAL*1 IDINF1,IDINF2,FILE4
      COMMON /CNTRL/ YSCALE,XSCALE,NPULSE,MANUAL,IRFLN,ICOFLN
     1,FILE 16),FILE2(16),FILE3(16),FILE4(16),AUTINC
      COMMON /ID/ IDINF1(72),IDINF2(72)
      COMMON /MUPROG/ INSTR(100)
      COMMON /CURSOR/ IDX1,RDY1,IDX2,RDY2,IWAIT,IDFSE1,IDFSE2
      COMMON /PLTLIM/ IFIRST,ILAST
      CNVT = 157.48031
      IF(AUTINC.NE.0)TYPE *,'AUTOINCREMENT IS ENABLED'
      IF(AUTINC.EQ.0)TYPE *,'AUTOINCREMENT IS DISABLED'
      IF(IRFLN.NE.0)TYPE *,'FILE NAMES WILL BE INDIVIDUALLY REQUESTED'
      IF(IRFLN.EQ.0)TYPE *,'FILE NAMES WILL NOT BE REQUESTED'
      IF(ICOFLN.NE.0)TYPE *,'FILE NAMES WILL BE ECHOED AFTER USE'
      IF(ICOFLN.EQ.0)TYPE *,'FILE NAMES WILL NOT BE ECHOED AFTER USE'
      IF(MANUAL.EQ.0)TYPE *,'PRESENTLY IN AUTOMATIC MODE'
      IF(MANUAL.EQ.1)TYPE *,'PRESENTLY IN MANUAL MODE'
      IF(IWAIT.EQ.0)
     1 TYPE *,'USER WILL NOT BE GIVEN CHANCE TO MOVE CURSOR'
      IF(IWAIT.EQ.1)TYPE *,'USER WILL BE ABLE TO MOVE CURSOR'
      TYPE *,'THE DATA X-SUBRANGE SELECTED IS:',IFIRST,ILAST
      TYPE *,'THE SERIES ID INFORMATION IS:'
      TYPE 101,(IDINF1(I),I=1,72)
      TYPE *, 'THE RUN INFORMATION IS:'
      TYPE 101,(IDINF2(I),I=1,72)
101   FORMAT(1X,72A1)
      PC = 1
100   ICH = INSTR(PC)
      PC=PC+1
      IF(ICH.EQ.0)RETURN
      GO TO (1,2,3,4,5,6,7,8,9,10,11,11,11,14,15,16,17
     1,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33
     2,34,36,37,38,39,40,41)ICH
      TYPE *, 'ERROR !!! UNIDENTIFIABLE OPCODE !!!'
      RETURN
1     TYPE *, 'ERASE BUFFER "A"'
      GO TO 100
2     TYPE *, 'ERASE BUFFER "B"'
      GO TO 100
3     TYPE *, 'ERASE BUFFER "D"'
      GO TO 100
4     TYPE *, 'ERASE ALL 3 DATA BUFFERS'
      GO TO 100
```

```
5       TYPE *, 'ACCUMULATE SINGLE PULSE TO "A"'
        GO TO 100
6       TYPE *, 'ACCUMULATE SINGLE PULSE TO "B"'
        GO TO 100
7       TYPE *, 'ACCUMULATE SINGLE PULSE TO "D"'
        GO TO 100
8       TYPE *, 'ACCUMULATE',INSTR(PC),' PULSES TO "A"'
        GO TO 35
9       TYPE *, 'ACCUMULATE',INSTR(PC),' PULSES TO "B"'
        GO TO 35
10      TYPE *, 'ACCUMULATE',INSTR(PC),' PULSES TO "D"'
        GO TO 35
11      IF(INSTR(PC).EQ.1)TYPE *, 'PLOT "A" ON SCREEN'
        IF(INSTR(PC).EQ.2)TYPE *, 'PLOT "B" ON SCREEN'
        IF(INSTR(PC).EQ.3)TYPE *, 'PLOT "D" ON SCREEN'
        PC=PC+1
        IF(ICH.EQ.11)GO TO 100
        IF(INSTR(PC).EQ.1)TYPE *, 'PLOT "A" ON SCREEN'
        IF(INSTR(PC).EQ.2)TYPE *, 'PLOT "B" ON SCREEN'
        IF(INSTR(PC).EQ.3)TYPE *, 'PLOT "D" ON SCREEN'
        PC=PC+1
        IF(ICH.EQ.12)GO TO 100
        IF(INSTR(PC).EQ.1)TYPE *, 'PLOT "A" ON SCREEN'
        IF(INSTR(PC).EQ.2)TYPE *, 'PLOT "B" ON SCREEN'
        IF(INSTR(PC).EQ.3)TYPE *, 'PLOT "D" ON SCREEN'
        PC=PC+1
        GO TO 100
14      TYPE *,'WRITE DATA IN BUFFER "A" TO FILE NUMBER ',INSTR(PC)
        GO TO 35
15      TYPE *,'WRITE DATA IN BUFFER "B" TO FILE NUMBER ',INSTR(PC)
        GO TO 35
16      TYPE *,'WRITE DATA IN BUFFER "D" TO FILE NUMBER ',INSTR(PC)
        GO TO 35
17      TYPE *,'READ DATA FROM FILE NUMBER ',INSTR(PC),' TO BUFFER "A"'
        GO TO 35
18      TYPE *,'READ DATA FROM FILE NUMBER ',INSTR(PC),' TO BUFFER "B"'
        GO TO 35
19      TYPE *,'READ DATA FROM FILE NUMBER ',INSTR(PC),' TO BUFFER "D"'
        GO TO 35
20      TYPE *,'WRITE AMPLITUDE/INTEGRAL DATA BUFFER TO FILE'
        GO TO 100
21      TYPE *,'ERASE CONTENTS OF AMPLITUDE/INTEGRAL DATA BUFFER'
        GO TO 100
22      TYPE *,'MOVE BURNER ',INSTR(PC),' STEPS OR '
        1,INSTR(PC)/CNVT,' M.M.'
        GO TO 35
23      TYPE *,'OPEN SHUTTER'
        GO TO 100
24      TYPE *,'CLOSE SHUTTER'
        GO TO 100
25      TYPE *,'READ DATA FROM FILE INTO AMPLITUDE/INTEGRAL DATA BUFFER'
        GO TO 100
26      TYPE *,'TYPE OUT CONTENTS OF AMPLITUDE,INTEGRAL DATA BUFFER'
        GO TO 100
```

```
27      TYPE *,'COMPUTE "D" BUFFER = "A" - "B" BUFFERS'
        GO TO 100
28      TYPE *,'INCREMENT FILENAME EXTENSION FOR FILENAME',INSTR(PC)
        GO TO 35
29      TYPE *,'INTEGRATE DISPLAYED CURVE BETWEEN LIMITS'
        GO TO 35
30      TYPE *,'MOVE LAST MEASUREMENT TAKEN TO A/I BUFFER'
        GO TO 100
31      TYPE *,'MEASURE AMPLITUDE FROM CURSOR POSITION'
        GO TO 100
32      TYPE *,'MEASURE AMPLITUDE FROM ACTUAL DATA SET DISPLAYED'
        GO TO 35
33      TYPE *,'WAIT FOR USER TO TYPE A CARRIAGE RETURN'
        GO TO 100
34      TYPE *,'ACQUIRE SCALE FACTORS FROM 7912 ONLY'
        GO TO 100
35      PC=PC+1
        GO TO 100
36      TYPE *,'MOVE DYE LASER GRATING ',INSTR(PC)
        1,' INCREMENTS OF .012 NM.'
        GO TO 35
37      GO TO (371,372,373)INSTR(PC)
371     TYPE *,'DO QUICK-N-DIRTY AMPLITUDE FROM DATA MIN, MAX'
        1,' BUFFER = A'
        GO TO 35
372     TYPE *,'DO QUICK-N-DIRTY AMPLITUDE FROM DATA MIN, MAX'
        1,' BUFFER = B'
        GO TO 35
373     TYPE *,'DO QUICK-N-DIRTY AMPLITUDE FROM DATA MIN, MAX'
        1,' BUFFER = D'
        GO TO 35
38      GO TO 100
39      TYPE *,'PLOT CONTENTS OF A/I BUFFER'
        GO TO 100
40      TYPE *,'TURN ON ION COLLECTOR H.V.'
        GO TO 100
41      TYPE *,'TURN OFF ION COLLECTOR H.V.'
        GO TO 100
        END
```

Now, the version for the liftim program:

```
        SUBROUTINE LISTPR
        INTEGER AUTINC,PC
        LOGICAL*1 FILE1,FILE2,FILE3,YSCALS,HSCALS,NUMSA
        LOGICAL*1 IDINF1,IDINF2,FILE4
        COMMON /CNTRL/ YSCALE,XSCALE,NPULSE,MANUAL,IRFLN,ICOFLN
        1,FILE1(16),FILE2(16),FILE3(16),FILE4(16),AUTINC,NWTPLT
        COMMON /ID/ IDINF1(72),IDINF2(72)
        COMMON /MUPROG/ INSTR(100),PC
        IF(AUTINC.NE.0)TYPE *,'AUTOINCREMENT IS ENABLED'
        COMMON /PLTLIM/ IFIRST,ILAST
        IF(AUTINC.EQ.0)TYPE *,'AUTOINCREMENT IS DISABLED'
```

```
      IF(IRFLN.NE.0)TYPE *,'FILE NAMES WILL BE INDIVIDUALLY REQUESTED'
      IF(IRFLN.EQ.0)TYPE *,'FILE NAMES WILL NOT BE REQUESTED'
      IF(ICOFLN.NE.0)TYPE *,'FILE NAMES WILL BE ECHOED AFTER USE'
      IF(ICOFLN.EQ.0)TYPE *,'FILE NAMES WILL NOT BE ECHOED AFTER USE'
      IF(MANUAL.EQ.0)TYPE *,'PRESENTLY IN AUTOMATIC MODE'
      IF(MANUAL.EQ.1)TYPE *,'PRESENTLY IN MANUAL MODE'
      IF(NWPLT.EQ.0)TYPE *,'A "RETURN" WILL BE AWAITED AFTER PLOTS'
      IF(NWPLT.EQ.1)TYPE *,'A "RETURN" WILL NOT BE AWAITED AFTER PLOTS'
      TYPE *,'THE DATA X-SUBRANGE SELECTED IS:',IFIRST,ILAST
      TYPE *,'THE SERIES ID INFORMATION IS:'
      TYPE 101,(IDINF1(I),I=1,72)
      TYPE *, 'THE RUN INFORMATION IS:'
      TYPE 101,(IDINF2(I),I=1,72)
101   FORMAT(1X,72A1)
      PC = 1
100   ICH = INSTR(PC)
      PC=PC+1
      IF(ICH.EQ.0)RETURN
      GO TO (1,2,3,4,5,6,7,8,9,10,11,11,11,14,15,16,17
     1,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33
     2,34,35,36,37,38)ICH
      TYPE *, 'ERROR !!! UNIDENTIFIABLE OPCODE !!!'
      RETURN
1     TYPE *, 'ERASE BUFFER "A"'
      GO TO 100
2     TYPE *, 'ERASE BUFFER "B"'
      GO TO 100
3     TYPE *, 'ERASE BUFFER "D"'
      GO TO 100
4     TYPE *, 'ERASE ALL 3 DATA BUFFERS'
      GO TO 100
5     TYPE *, 'ACCUMULATE SINGLE PULSE TO "A"'
      GO TO 100
6     TYPE *, 'ACCUMULATE SINGLE PULSE TO "B"'
      GO TO 100
7     TYPE *, 'ACCUMULATE SINGLE PULSE TO "D"'
      GO TO 100
8     TYPE *, 'ACCUMULATE',INSTR(PC),' PULSES TO "A"'
      GO TO 35
9     TYPE *, 'ACCUMULATE',INSTR(PC),' PULSES TO "B"'
      GO TO 35
10    TYPE *, 'ACCUMULATE',INSTR(PC),' PULSES TO "D"'
      GO TO 35
11    IF(INSTR(PC).EQ.1)TYPE *, 'PLOT "A" ON SCREEN'
      IF(INSTR(PC).EQ.2)TYPE *, 'PLOT "B" ON SCREEN'
      IF(INSTR(PC).EQ.3)TYPE *, 'PLOT "D" ON SCREEN'
      PC=PC+1
      IF(ICH.EQ.11)GO TO 100
      IF(INSTR(PC).EQ.1)TYPE *, 'PLOT "A" ON SCREEN'
      IF(INSTR(PC).EQ.2)TYPE *, 'PLOT "B" ON SCREEN'
      IF(INSTR(PC).EQ.3)TYPE *, 'PLOT "D" ON SCREEN'
      PC=PC+1
      IF(ICH.EQ.12)GO TO 100
      IF(INSTR(PC).EQ.1)TYPE *, 'PLOT "A" ON SCREEN'
```

```
      IF(INSTR(PC).EQ.2)TYPE *, 'PLOT "B" ON SCREEN'
      IF(INSTR(PC).EQ.3)TYPE *, 'PLOT "D" ON SCREEN'
      PC=PC+1
      GO TO 100
14    TYPE *,'WRITE DATA IN BUFFER "A" TO FILE NUMBER ',INSTR(PC)
      GO TO 35
15    TYPE *,'WRITE DATA IN BUFFER "B" TO FILE NUMBER ',INSTR(PC)
      GO TO 35
16    TYPE *,'WRITE DATA IN BUFFER "D" TO FILE NUMBER ',INSTR(PC)
      GO TO 35
17    TYPE *,'READ DATA FROM FILE NUMBER ',INSTR(PC),' TO BUFFER "A"'
      GO TO 35
18    TYPE *,'READ DATA FROM FILE NUMBER ',INSTR(PC),' TO BUFFER "B"'
      GO TO 35
19    TYPE *,'READ DATA FROM FILE NUMBER ',INSTR(PC),' TO BUFFER "D"'
      GO TO 35
20    TYPE *,'WRITE AMPLITUDE/INTEGRAL DATA BUFFER TO FILE'
      GO TO 100
21    TYPE *,'ERASE CONTENTS OF AMPLITUDE/INTEGRAL DATA BUFFER'
      GO TO 100
22    GO TO 35
23    GO TO 100
24    GO TO 100
25    TYPE *,'READ DATA FROM FILE INTO AMPLITUDE/INTEGRAL DATA BUFFER'
      GO TO 100
26    TYPE *,'TYPE OUT CONTENTS OF AMPLITUDE,INTEGRAL DATA BUFFER'
      GO TO 100
27    TYPE *,'COMPUTE "D" BUFFER = "A" - "B" BUFFERS'
      GO TO 100
28    TYPE *,'INCREMENT FILENAME EXTENSION FOR FILENAME',INSTR(PC)
      GO TO 35
29    TYPE *,'INTEGRATE DISPLAYED CURVE BETWEEN LIMITS'
      GO TO 35
30    TYPE *,'MOVE LAST MEASUREMENT TAKEN TO A/I BUFFER'
      GO TO 100
31    TYPE *,'MEASURE AMPLITUDE FROM CURSOR POSITION'
      GO TO 100
32    TYPE *,'MEASURE AMPLITUDE FROM ACTUAL DATA SET DISPLAYED'
      GO TO 35
33    TYPE *,'WAIT FOR USER TO TYPE A CARRIAGE RETURN'
      GO TO 100
34    TYPE *,'ACQUIRE SCALE FACTORS FROM 7912 ONLY'
      GO TO 100
35    PC=PC+1
      GO TO 100
36    GO TO (361,362,363)INSTR(PC)
361   TYPE *,'DO QUICK-N-DIRTY AMPLITUDE FROM DATA MIN, MAX'
      1,' BUFFER = A'
      GO TO 35
362   TYPE *,'DO QUICK-N-DIRTY AMPLITUDE FROM DATA MIN, MAX'
      1,' BUFFER = B'
      GO TO 35
363   TYPE *,'DO QUICK-N-DIRTY AMPLITUDE FROM DATA MIN, MAX'
      1,' BUFFER = D'
```

```
        GO TO 35
37      TYPE *,'CHAIN TO FITTING PROGRAM'
        GO TO 100
38      TYPE *,'PLOT CONTENTS OF A/I BUFFER'
        GO TO 100
        RETURN
        END
```

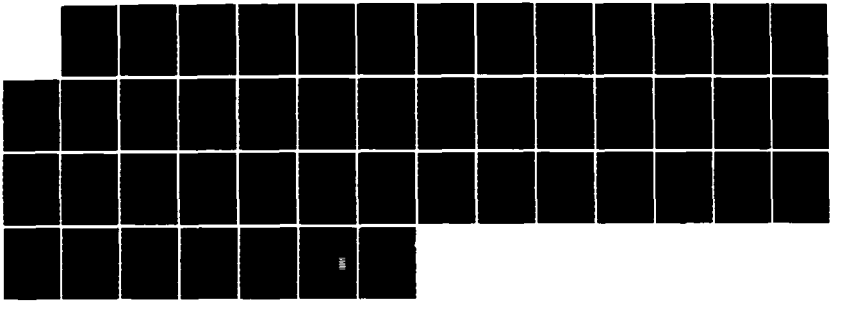
The next major set of routines that the main routines of the programs call are the setup mode routines. These routines are used to set the various microprogram independent flags and states that the system needs in order to operate with its various features. First, the version for the multiphoton program:

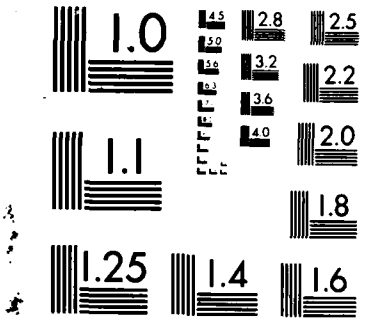
```
        SUBROUTINE SETUP
        INTEGER DATA,AUTINC,SELECT,MANUAL
        LOGICAL*1 FILE1,FILE2,FILE3,YSCALS,HSCALS,NUMSA
        LOGICAL*1 IDINF1,IDINF2,FILE4
        COMMON /CNTRL/ YSCALE,XSCALE,NPULSE,MANUAL,IRFN,ICOFLN
        1,FILE1(16),FILE2(16),FILE3(16),FILE4(16),AUTINC
        COMMON /ID/ IDINF1(72),IDINF2(72)
        COMMON /DATA/ DATA(512)
        COMMON /VSCALF/ YSCALS(20)
        COMMON /HSCALF/ HSCALS(20)
        COMMON /BUFFA/ RADATA(512)
        COMMON /CURSOR/ IDX1,RDY1,IDX2,RDY2,IWAIT,IDFSE1,IDFSE2
        COMMON /PLTLIM/ IFIRST,ILAST
        EXTERNAL DUMP,GETSCF
        DATA IY /'Y'/

C   Put up headers, selection menu
        TYPE 100
11      TYPE 101
        TYPE 102
        TYPE 103
C  Get selection, go to appropriate code
        ACCEPT 104,SELECT
        GO TO (1,2,3,4,5,8,9,12)SELECT
        GO TO 11
C  Get all the I.D. information
1       TYPE 105
        ACCEPT 106, (IDINF1(I),I=1,72)
        TYPE 107
        ACCEPT 106, (IDINF2(I),I=1,72)
        GO TO 11
C  Get the first and last plotting channels for the data
2       TYPE 120,IFIRST,ILAST
        ACCEPT 121, I1,I2
        IF(I1.EQ.0.OR.I2.EQ.0)GO TO 11
        IFIRST = I1
        ILAST = I2
        GO TO 11
```

```
C  Get the 4 file names for file accesses
3       TYPE 112
        CALL GETNAM(FILE1)
        TYPE 113
        CALL GETNAM(FILE2)
        TYPE 114
        CALL GETNAM(FILE3)
        TYPE 115
        CALL GETNAM(FILE4)
        GO TO 11
C  Set/clear the autoincrement flag
4       AUTINC = 0
        TYPE 109
        ACCEPT 110,ISEL
        IF (ISEL .NE. IY)GO TO 11
        IRFN = 0
        AUTINC = 1
        GO TO 11
C  Get the new value of the A/I buffer pointer, and clear the
C  buffer above it.
5       TYPE 122
        ACCEPT 104,IPTR
        DO 51 I = IPTR,100
        AIVALS(I) = 0
51      CONTINUE
        GO TO 11
C  Set/clear the echo file names flag
8       TYPE 118
        ICOFLN = 0
        ACCEPT 110,ISEL
        IF (ISEL .EQ. IY)ICOFLN = 1
        GO TO 11
C  Set/clear the wait for user cursor input flag
9       TYPE 119
        IWAIT = 1
        ACCEPT 110,ISEL
        IF(ISEL.EQ.IY.AND.IDFSE1.EQ.0.AND.IDFSE2.EQ.0)GO TO 15
        IF(ISEL.EQ.IY)IWAIT = 0
        GO TO 11
15      TYPE *,'WARNING!! YOU MUST INITIALIZE CURSOR LIMITS'
        1,' BEFORE DOING THIS'
        IWAIT = 1
        GO TO 11
12      RETURN

C***************** FORMAT STATEMENTS FOR ROUTINE SETSER *********************
100     FORMAT(1X'SETUP FOR SERIES MODE ...'//)
101     FORMAT(1X'SERIES SETUP MENU :'/)
102     FORMAT(
        1 10X'1 = ENTER ID INFORMATION FOR SERIES AND RUN'/
        2,10X'2 = SET START AND END CHANNELS FOR DATA SUBSET'/
        3,10X'3 = ENTER THE 4 FILE NAMES FOR FILE ACCESS'/
        4,10X'4 = SET/CLEAR AUTOINCREMENT OF FILENAME FEATURE'/
        5,10X'5 = RESET THE A/I BUFFER POINTER FOR ERROR CORRECTION'/
```

AD-A173 903 AN AUTOMATED SYSTEM FOR THE CONTROL OF AND DATA ACQUISITION FROM MULTIPHO..(U) ARMY BALLISTIC RESEARCH LAB ABERDEEN PROVING GROUND MD M A DEWILDE SEP 86 2/2

UNCLASSIFIED BRL-TR-2759 F/G 9/2 NL





MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

```
     8,10X'6 = SET/CLEAR ECHO FILE NAME FEATURE'/
     9,10X'7 = SET/CLEAR WAIT FOR CURSOR POSITION ACCEPT FLAG'/
     9,10X'8 = RETURN TO MAIN SELECT MENU'//)

103  FORMAT('$',15X,'SELECTION ? >')
104  FORMAT(I4)
105  FORMAT(1X'TYPE IN THE SERIES ID INFORMATION, 72 CHARACTERS MAX'/)
106  FORMAT(72A1)
107  FORMAT(1X'TYPE IN THE RUN ID INFORMATION, 72 CHARACTERS MAX'/)
109  FORMAT('$THE AUTOINCREMENT FEATURE IS NOW DISABLED.'
     1,' ENABLE ? (Y OR N)>')
110  FORMAT(A2)
112  FORMAT(1X'FOR FILE NAME #1,')
113  FORMAT(1X'FOR FILE NAME #2,')
114  FORMAT(1X'FOR FILE NAME #3,')
115  FORMAT(1X'FOR THE AMPLITUDE/INTEGRAL DATA FILE NAME,')
116  FORMAT(1X'VERTICAL SCALE = ',E8.2,' V/DIV',/
     1,1X'HORIZONTAL SCALE = ',E8.2,' T/DIV'/)
118  FORMAT('$THE ECHO FILE NAME FEATURE IS NOW DISABLED.'
     1,' ENABLE ? (Y OR N)>')
119  FORMAT('$THE CURSOR ACCEPTANCE BY USER WILL BE AWAITED.'
     1' CHANGE? (Y OR N)>')
120  FORMAT('$ENTER SUBRANGE ENDPOINTS (',I3,' ,',I3,') >')
121  FORMAT(2I3)
122  FORMAT('$TYPE IN THE NEW VALUE FOR THE A/I POINTER >>')
C*********************************************************************************
```

The corresponding code for the LIFTIM program is:

```
     SUBROUTINE SETUP
     INTEGER DATA,AUTINC,SELECT,MANUAL
     LOGICAL*1 FILE1,FILE2,FILE3,YSCALS,HSCALS,NUMSA
     LOGICAL*1 IDINF1,IDINF2,FILE4
     COMMON /CNTRL/ YSCALE,XSCALE,NPULSE,MANUAL,IRFN,ICOFLN
     1,FILE1(16),FILE2(16),FILE3(16),FILE4(16),AUTINC,NWTPLT
     COMMON /ID/ IDINF1(72),IDINF2(72)
     COMMON /DATA/ DATA(512)
     COMMON /VSCALF/ YSCALS(20)
     COMMON /HSCALF/ HSCALS(20)
     COMMON /BUFFA/ RADATA(512)
     COMMON /PLTLIM/ IFIRST,ILAST
     COMMON /AI/ AIX,AIY,AIVALS(100),IPTR
     EXTERNAL DUMP,GETSCF
     DATA IY /'Y'/

     TYPE 100
11   TYPE 101
     TYPE 102
     TYPE 103
     ACCEPT 104,SELECT
     GO TO (1,2,3,4,5,6,8,14,12)SELECT
     GO TO 11
```

```
C   Get I.D. information
1       TYPE 105
        ACCEPT 106, (IDINF1(I),I=1,72)
        TYPE 107
        ACCEPT 106, (IDINF2(I),I=1,72)
        GO TO 11
C   Get plotting range for X values
2       TYPE 120,IFIRST,ILAST
        ACCEPT 121, I1,I2
        IF(I1.EQ.0.OR.I2.EQ.0)GO TO 11
        IFIRST = I1
        ILAST = I2
        GO TO 11
C   Get all 4 file names to be used
3       TYPE 112
        CALL GETNAM(FILE1)
        TYPE 113
        CALL GETNAM(FILE2)
        TYPE 114
        CALL GETNAM(FILE3)
        TYPE 115
        CALL GETNAM(FILE4)
        GO TO 11
C   Set/clear autoincrement flag
4       AUTINC = 0
        TYPE 109
        ACCEPT 110,ISEL
        IF (ISEL .NE. IY)GO TO 11
        IRFN = 0
        AUTINC = 1
        GO TO 11
C   Set A/I buffer pointer to new value, clear buffer above it.
5       TYPE 122
        ACCEPT 104,IPTR
        DO 51 I = IPTR,100
        AIVALS(I) = 0
51      CONTINUE
        GO TO 11
C   Do sample digitization and display
6       CALL CIF(DUMP)
        CALL CIF(GETSCF)
        CALL CVTSCF
        DO 10 I=1,512
        RADATA(I)=DATA(I)
10      CONTINUE
C  PLOT 1 FILE, IN POSITION 1, FROM BUFFER 1 (1,1,1)
        CALL MAKPLT(1,1,1)
        TYPE 116,YSCALE,XSCALE
        GO TO 11
C   Set/clear echo file name flag
8       TYPE 118
        ICOFLN = 0
        ACCEPT 110,ISEL
        IF (ISEL .EQ. IY)ICOFLN = 1
```

```
        GO TO 11
C   Set/clear flag to await a carriage return after each plot
14      NWTPLT = 0
        TYPE 119
        ACCEPT 110,ISEL
        IF (ISEL.EQ.IY)NWTPLT=1
        GO TO 11
12      RETURN

C****************** FORMAT STATEMENTS FOR ROUTINE SETSER ********************
100     FORMAT(1X'SETUP FOR SERIES MODE ...'//)
101     FORMAT(1X'SERIES SETUP MENU :'/)
102     FORMAT(
        1 10X'1 = ENTER ID INFORMATION FOR SERIES AND RUN'/
        2,10X'2 = SET START AND END CHANNELS FOR DATA SUBSET'/
        3,10X'3 = ENTER THE 4 FILE NAMES FOR FILE ACCESS'/
        4,10X'4 = SET/CLEAR AUTOINCREMENT OF FILENAME FEATURE'/
        5,10X'5 = RESET THE A/I BUFFER POINTER FOR ERROR CORRECTION'/
        6,10X'6 = CHECK DIGITIZER CONNECTIONS,GET SCALE FACTORS'/
        8,10X'7 = SET/CLEAR ECHO FILE NAME FEATURE'/
        8,10X'8 = SET/CLEAR WAIT AFTER PLOTS FEATURE'/
        9,10X'9 = RETURN TO MAIN SELECT MENU'//)
103     FORMAT('$',15X,'SELECTION ? >')
104     FORMAT(I4)
105     FORMAT(1X'TYPE IN THE SERIES ID INFORMATION, 72 CHARACTERS MAX'/)
106     FORMAT(72A1)
107     FORMAT(1X'TYPE IN THE RUN ID INFORMATION, 72 CHARACTERS MAX'/)
109     FORMAT('$THE AUTOINCREMENT FEATURE IS NOW DISABLED.'
        1,' ENABLE ? (Y OR N)>')
110     FORMAT(A2)
112     FORMAT(1X'FOR FILE NAME #1,')
113     FORMAT(1X'FOR FILE NAME #2,')
114     FORMAT(1X'FOR FILE NAME #3,')
115     FORMAT(1X'FOR THE AMPLITUDE/INTEGRAL DATA FILE NAME,')
116     FORMAT(1X'VERTICAL SCALE = ',E8.2,' V/DIV',/
        1,1X'HORIZONTAL SCALE = ',E8.2,' T/DIV'/)
118     FORMAT('$THE ECHO FILE NAME FEATURE IS NOW DISABLED.'
        1,' ENABLE ? (Y OR N)>')
119     FORMAT('$A "RETURN" WILL BE AWAITED AFTER PLOTS.'
        1,' CHANGE? (Y OR N)>')
120     FORMAT('$ENTER SUBRANGE ENDPOINTS (',I3,' ,',I3,') >')
121     FORMAT(2I3)
122     FORMAT('$TYPE IN THE NEW VALUE FOR THE A/I POINTER >>')
C*******************************************************************************

        END
```

The remaining code in the Lifetime program is that which accomplishes the chaining. In order not to have any discontinuities in the flow of the program, a dummy routine is also given that simulates the program being chained into, and illustrates the method of setting such routines up. In order to formalize the process, an on-line documentation file was created, and follows:

Documentation file for chaining interface with the Lifetime Data Acquisition program 14-NOV-83

I. Setting up the program to be chained into from "LIFTIM"

The Program MUST be Named "EXPFIT.SAV", or have the call changed in the liftime program itself to reflect the new name. The first executable statement in the program to be chained into must be:

```
        CALL RCHAIN(IFLAG,IVAR,0)
```

If the value of IFLAG is -1, then the program has been chained into. The data for the program to work on is in a chaining file. At the completion of this program, a return chain must be followed in order to return to the original program. If the value is not -1, then the program has been entered via the "run" command and is to be used stand-alone. The normal FORTRAN call exit or stop is used to terminate the program. A typical first section of code would look like:

```
        CALL RCHAIN(IFLAG,IVAR,0)
        IF (IFLAG.NE.-1)GO TO 10
C   HERE WOULD GO CODE TO READ IN THE DESIRED DATA FROM THE CHAIN DATA FILE:
C   SY:CHNFIL.LIF
C
C
        GO TO 20

10      CONTINUE
C  HERE WOULD GO THE CODE TO READ IN THE NORMAL DATA FILES FOR STAND-ALONE USE
C  FROM USER-SUPPLIED FILE NAMES.
C

20      CONTINUE
C  HERE WOULD BE THE ENTRY POINT FOR WHATEVER THE PROGRAM WOULD DO FOR EITHER
C  FORM OF PROGRAM ENTRY
```

In this way, the program may be chained into from another routine, or can be run as a stand-alone program, transparent to the user.

II. The following is the format of the chain data file containing everything known of the status of the lifetime data program at the time of chaining. The file

consists of 1 record, 3498 words long, unformatted binary. Typical code to open the file would be:

```
INTEGER AUTINC,SELECT,MANUAL,PC
BYTE FILE1,FILE2,FILF3,IDINF1,IDINF2,FILE4
DIMENSION RADATA(512),RBDATA(512),RDDATA(512)
DIMENSION FILE1(16),FILE2(16),FILE3(16),FILE4(16)
DIMENSION IDINF1(72),IDINF2(72),INSTR(100),AIVALS(100)

CALL ASSIGN(9,'SY:CHNFIL.LIF',0,'OLD')
DEFINE FILE 9 (1,3498,U,IVAR)
READ (9'1) (RADATA(I),I=1,512),(RBDATA(I),I=1,512)
2,(RDDATA(I),I=1,512),YSCALE,XSCALE,NPULSE,MANUAL,IRFLN,ICOFLN
3,(FILE1(I),I=1,16),(FILE2(I),I=1,16),(FILE3(I),I=1,16)
4,(FILE4(I),I=1,16),AUTINC,(IDINF1(I),I=1,72)
5,(IDINF2(I),I=1,72),(INSTR(I),I=1,100),PC,AIX
6,AIY,(AIVALS(I),I=1,100),IPTR,SCALF,YMIN,YMAX,NUMREM
CALL CLOSE(9)
```

WHERE RADATA,RBDATA and RDDATA are the contents of buffers A,B, and D respectively. YSCALE and XSCALE are the vertical and horizontal scale factors, in v/div and sec/div, where a division is 51.2 channels of the data (512 channels = 10 divisions). AIVALS is an array of data values to receive measurements for later processing. IPTR points to the next available unused element of this array and is incremented after the element is filled. Other variables are needed by the liftim program to pick up where it left off, and are of little use to external programs. Of course unneeded variables are read to dummy variables.

III. Procedure for exiting from the chained into auxiliary program

In place of the normal FORTRAN "stop" or "call exit" statement, the following code is to be placed:

```
DIMENSION SPEC(2)
DATA SPEC/6RSY LIF,6RTIMSAV/

IF(IFLAG.EQ.-1)CALL CHAIN(SPEC,IVAR,0)
IF(IFLAG.NE.-1)CALL EXIT
END
```

IV. Passing Data Values To The Lifetime Program

Probably the easiest way to do this is through the AIVALS array in the chain file. Usually data that have the form of 2-tuples are placed in this array in sequential pairs, while individual data are placed sequentially. Each time a datum is placed in the array, IPTR *MUST* be updated to point to the next free space. Both the array *AND IPTR* must be re-written to the chain file if they are to be plotted, etc. by the support library of routines that go with LIFTIM.

********** End Of Documentation File For Chain Interface **********

The actual routines used by the lifetime program are:

```
      SUBROUTINE CHNFIT(NUMREM)
C  This routine stores the software context of the lifetime program, and
C  chains to another program named "EXPFIT.SAV"
      DIMENSION SPEC(2)
      DATA SPEC/6RSY EXP,6RFITSAV/
      CALL STORE(NUMREM)
      CALL CHAIN(SPEC,IVAR,0)
      END


      SUBROUTINE STORE(NUMREM)
C   This routine creates the file "CHNFIL.LIF", and stores the entire
C   software context of the program in it.
      INTEGER AUTINC,SELECT,MANUAL,PC
      LOGICAL*1 FILE1,FILE2,FILE3,IDINF1,IDINF2,FILE4
      COMMON /BUFFA/ RADATA(512)
      COMMON /BUFFB/ RBDATA(512)
      COMMON /BUFFD/ RDDATA(512)
      COMMON /CNTRL/ YSCALE,XSCALE,NPULSE,MANUAL,IRFLN,ICOFLN
      1,FILE1(16),FILE2(16),FILE3(16),FILE4(16),AUTINC,NWTPLT
      COMMON /ID/ IDINF1(72),IDINF2(72)
      COMMON /MUPROG/ INSTR(100),PC
      COMMON /AI/ AIX,AIY,AIVALS(100),IPTR
      COMMON /PLT/ SCALF,YMIN,YMAX
      COMMON /PLTLIM/ IFIRST,ILAST

      CALL ASSIGN(9,'SY:CHNFIL.LIF',0,'NEW')
      DEFINE FILE 9 (1,3501,U,IVAR)
      WRITE (9'1) (RADATA(I),I=1,512),(RBDATA(I),I=1,512)
      2,(RDDATA(I),I=1,512),YSCALE,XSCALE,NPULSE,MANUAL,IRFLN,ICOFLN
      3,(FILE1(I),I=1,16),(FILE2(I),I=1,16),(FILE3(I),I=1,16)
      4,(FILE4(I),I=1,16),AUTINC,(IDINF1(I),I=1,72),(IDINF2(I),I=1,72)
      5,(INSTR(I),I=1,100),PC,AIX,AIY,(AIVALS(I),I=1,100),IPTR,SCALF
      6,YMIN,YMAX,NUMREM,NWTPLT,IFIRST,ILAST
      CALL CLOSE(9)
      RETURN
      END


      SUBROUTINE RESTOR(NUMREM)
C  This routine opens the file "CHNFIL.LIF", and restores the program
C  context with the data read in therefrom.

      INTEGER AUTINC,SELECT,MANUAL,PC
      LOGICAL*1 FILE1,FILE2,FILE3,IDINF1,IDINF2,FILE4
      COMMON /BUFFA/ RADATA(512)
      COMMON /BUFFB/ RBDATA(512)
      COMMON /BUFFD/ RDDATA(512)
      COMMON /CNTRL/ YSCALE,XSCALE,NPULSE,MANUAL,IRFLN,ICOFLN
      1,FILE1(16),FILE2(16),FILE3(16),FILE4(16),AUTINC,NWTPLT
```

```
      COMMON /ID/ IDINF1(72),IDINF2(72)
      COMMON /MUPROG/ INSTR(100),PC
      COMMON /AI/ AIX,AIY,AIVALS(100),IPTR
      COMMON /PLT/ SCALF,YMIN,YMAX
      COMMON /PLTLIM/ IFIRST,ILAST

C  open the old file for reading...
      CALL ASSIGN(9,'SY:CHNFIL.LIF',0,'OLD')
      DEFINE FILE 9 (1,3501,U,IVAR)
C  read in all the data...
      READ (9'1) (RADATA(I),I=1,512),(RBDATA(I),I=1,512)
     2,(RDDATA(I),I=1,512),YSCALE,XSCALE,NPULSE,MANUAL,IRFLN,ICOFLN
     3,(FILE1(I),I=1,16),(FILE2(I),I=1,16),(FILE3(I),I=1,16)
     4,(FILE4(I),I=1,16),AUTINC,(IDINF1(I),I=1,72),(IDINF2(I),I=1,72)
     5,(INSTR(I),I=1,100),PC,AIX,AIY,(AIVALS(I),I=1,100),IPTR,SCALF
     6,YMIN,YMAX,NUMREM,NWTPLT,IFIRST,ILAST
C  and close it...
      CALL CLOSE(9)
      RETURN
      END


      SUBROUTINE CHNLIF
C  this routine causes a chain to a program named "LIFTIM.SAV" to occur.
C  This is the return chain to the original lifetime program.
      DIMENSION SPEC(2)
      DATA SPEC/6RSY LIF ,6RTIMSAV/
      CALL CHAIN(SPEC,IVAR,0)
      END
```

This is the dummy program that ensures continuity of operation of liftime, in the absence of a real program EXPFIT:

```
      PROGRAM EXPFIT
      CALL RCHAIN(IVAL,IVAR,0)
      IF(IVAL.EQ.-1)TYPE *,'EXPFIT CHAINED INTO DETECTED'
      CALL CHNLIF
      END
```

V. THE ACCESSORY A/I DATA FILE PLOTTING PROGRAM

A plotting program was written in order to generate publication quality hard copy plots from data that was written into files from the A/I buffer. The hardware device that performs the plotting is a Versatec model D1200 electrostatic printer/plotter, and is driven by that company's plotting software described in Reference 5. The text of the plotting program follows:

---

[5]Operational Design Manual for Versaplot Software, Part No. 50001 - 90001, Versatec, Inc., Santa Clara, CA, 1974.

```
C       MIZPLT.FOR      6-SEP-83
C
C       LINK VIA:
C
C       .R LINK
C       *MIZPLT=MIZPLT,LSQUAR,SYSLIB/F/C
C       *MODE,NOTE,FORM,AXES,VPLIB,TCSLIB
C
C       PROGRAM TO PLOT X,Y DATA POINTS
C        --ONE TO FIVE CASES
C       AND A LEAST SQUARES FIT OF THE DATA


        BYTE FILNAM,IDINF1,IDINF2
        DIMENSION XD(200,5),YD(200,5),
        1 XP(200,5),YP(200,5),XTEXT(6),YTEXT(6),
        2 NPTS(5),NP(5),
        3 ISYM(5),AIVALS(200),FILNAM(16),IDINF1(72),IDINF2(72)
        DATA X/'X'/,Y/'Y'/,HIGH/.1/,WIDE/.067/
        DATA IY /'Y'/
        INTEGER ORDF
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
C  RESERVE ENOUGH CHANNELS FOR VERSATEC STUFF
        DO 5 J=1,4
        I=IGETC()
5       CONTINUE
C  CLEAR DATA ARRAYS
10      DO 20 J=1,5
        DO 20 I=1,200
        XD(I,J)=0
        YD(I,J)=0
20      CONTINUE
        DO 30 J=1,5
        DO 30 I=1,200
        XP(I,J)=0
        YP(I,J)=0
30      CONTINUE
C  CLEAR SCREEN
        CALL NEWPAG
C GET NO. OF CASES
50      TYPE 902
        ACCEPT 901,NCASE
        IF(NCASE.LT.1 .OR. NCASE.GT.5) GOTO 50
C SEE IF 1 OR 2 FILES TO PLOT
        TYPE 201
        ACCEPT 202, NUMINP
201     FORMAT('$NUMBER OF INPUT FILES ? (1 OR 2) >')
202     FORMAT(I4)
C GET THE FILE NAME FOR THE INPUT DATA
        CALL GETNAM(FILNAM)
C OPEN THE FILE FOR INPUT
        OPEN(UNIT=9,NAME=FILNAM,ACCESS='DIRECT',INITIALSIZE=2,
        1RECORDSIZE=256,TYPE='OLD')
```

```
C READ IN THE FILE
        READ(9'1) (AIVALS(I),I=1,100),(IDINF1(I),I=1,72)
        1,(IDINF2(I),I=1,72)
        CALL CLOSE(9)
C  FIND FIRST ZERO DATA VALUE IN AIVALS
        I = 0
203     I = I + 1
        IF(AIVALS(I).NE.0)GO TO 203
C  DO NEXT FILE IF CHOSEN
        IF(NUMINP.EQ.1)GO TO 204
        CALL GETNAM(FILNAM)
        OPEN(UNIT=9,NAME=FILNAM,ACCESS='DIRECT',INITIALSIZE=2,
        1RECORDSIZE=256,TYPE='OLD')
        READ(9'1) (AIVALS(J),J=I,I+100)
        CALL CLOSE(9)
C GET NUMBER OF DATA IN DATA GROUP
204     TYPE 101
101     FORMAT('$TYPE IN THE NUMBER OF DATA IN A DATA GROUP >')
        ACCEPT 901, NUMGRP
C GET THE STARTING X-VALUE
        TYPE 103
103     FORMAT('$TYPE IN THE STARTING X VALUE >')
        ACCEPT 104,XLOW
104     FORMAT(E15.0)
C GET THE X-INCREMENT AMOUNT
        TYPE 105
105     FORMAT('$TYPE IN THE X-DATA INCREMENT >')
        ACCEPT 104,XINCR
C INPUT DATA
        DO 80 J=1,NCASE
        CALL NEWPAG
C GET POSITION IN DATA GROUP OF DATUM
        TYPE 102
102     FORMAT('$TYPE IN THE POSITION OF THE DATA IN THE DATA GROUP >')
        ACCEPT 901, NUMPOS
        IPOS=NUMPOS
        XDATA = XLOW
        DO 60 I=1,200
        XD(I,J)= XDATA
        YD(I,J) = AIVALS(IPOS)
        XDATA = XDATA + XINCR
        IPOS = IPOS + NUMGRP
        IF(YD(I,J).EQ.0) GO TO 70
60      CONTINUE
70      NPTS(J)=I-1
        CALL SIZSEL(ICHAR,HIGH,WIDE)
        ISYM(J)=ICHAR
80      CONTINUE
C GET PLOT SCALES
100     CALL NEWPAG
        TYPE 106
106     FORMAT(1X'THE PLOT LABELS WILL BE:')
        TYPE 107,(IDINF1(I),I=1,    I = 0
203     I = I + 1
```

```
        IF(AIVALS(I).NE.0)GO TO 203
C  DO NEXT FILE IF CHOSEN
        IF(NUMINP.EQ.1)GO TO 204
        CALL GETNAM(FILNAM)
        OPEN(UNIT=9,NAME=FILNAM,ACCESS='DIRECT',INITIALSIZE=2,
        1RECORDSIZE=256,TYPE='OLD')
        READ(9'1) (AIVALS(J),J=I,I+100)
        CALL CLOSE(9)
C GET NUMBER OF DATA IN DATA GROUP
204     TYPE 101
101     FORMAT('$TYPE IN THE NUMBER OF DATA IN A DATA GROUP >')
        ACCEPT 901, NUMGRP
C GET THE STARTING X-VALUE
        TYPE 103
103     FORMAT('$TYPE IN THE STARTING X VALUE >')
        ACCEPT 104,XLOW
104     FORMAT(E15.0)
C GET THE X-INCREMENT AMOUNT
        TYPE 105
105     FORMAT('$TYPE IN THE X-DATA INCREMENT >')
        ACCEPT 104,XINCR
C INPUT DATA
        DO 80 J=1,NCASE
        CALL NEWPAG
C GET POSITION IN DATA GROUP OF DATUM
        TYPE 102
102     FORMAT('$TYPE IN THE POSITION OF THE DATA IN THE DATA GROUP >')
        ACCEPT 901, NUMPOS
        IPOS=NUMPOS
        XDATA = XLOW
        DO 60 I=1,200
        XD(I,J)= XDATA
        YD(I,J) = AIVALS(IPOS)
        XDATA = XDATA + XINCR
        IPOS = IPOS + NUMGRP
        IF(YD(I,J).EQ.0) GO TO 70
60      CONTINUE
70      NPTS(J)=I-1
        CALL SIZSEL(ICHAR,HIGH,WIDE)
        ISYM(J)=ICHAR
80      CONTINUE
C GET PLOT SCALES
100     CALL NEWPAG
        TYPE 106
106     FORMAT(1X'THE PLOT LABELS WILL BE:')
        TYPE 107,(IDINF1(I),I=1,72)
        TYPE 107,(IDINF2(I),I=1,72)
107     FORMAT(1X,72A1)
        TYPE 108
108     FORMAT('$ACCEPT THE FIRST? (Y OR N)>')
        ACCEPT 109,INPA
109     FORMAT(A2)
        IF (INPA.EQ.IY)GO TO 35
        TYPE 916,1
```

```
916     FORMAT(' INPUT PLOT LABEL ',I2)
917     FORMAT(72A1)
        ACCEPT 917,(IDINF1(I),I=1,72)
35      TYPE 114
114     FORMAT('$ACCEPT THE SECOND? (Y OR N) >')
        ACCEPT 109, INPA
        IF (INPA.EQ.IY) GO TO 36
        TYPE 916,2
        ACCEPT 917,(IDINF2(I),I=1,72)
36      CALL MINMAX(XMIN,XMAX,XD,NCASE)
        TYPE 906,XMIN,XMAX
        TYPE 111
111     FORMAT('$  ACCEPT X-LIMITS? (Y OR N)>')
        ACCEPT 109,INPA
        IF(INPA.EQ.IY)GO TO 37
        TYPE 113
113     FORMAT('$TYPE IN THE NEW MIN AND MAX >>')
        ACCEPT 905,XMIN,XMAX
37      CALL MINMAX(YMIN,YMAX,YD,NCASE)
        TYPE 908,YMIN,YMAX
        TYPE 112
112     FORMAT('$  ACCEPT Y-LIMITS? (Y OR N)>')
        ACCEPT 109,INPA
        IF(INPA.EQ.IY) GO TO 38
        TYPE 113
        ACCEPT 905,YMIN,YMAX
C INPUT AXES LABELS
38      TYPE 918,X
        ACCEPT 919,(XTEXT(I),I=1,6)
        TYPE 918,Y
        ACCEPT 919,(YTEXT(I),I=1,6)
C CALCULATE SCALES
##C  )SE(6)
C PLOT DATA
        S=9999.
        CALL MODE(1,.833,1.0,0.0)
C SET X PLOT LIMITS
        CALL MODE(2,5.728,-.312,.416)
C SET Y PLOT LIMITS
        CALL MODE(3,6.1,-1.66,0.75)
C SET PLOT SIZE
        CALL MODE(7,XLEN,YLEN,S)
C SET X SCALE - MANUAL
        CALL MODE(8,XMIN,XSCALE,S)
C SET Y SCALE - MANUAL
        CALL MODE(9,YMIN,YSCALE,S)
        CALL AXES(16.3,XTEXT,20.3,YTEXT)
C DRAW BOX AROUND PLOT
        CALL FORM(1,XLEN,1,YLEN)
C DRAW PLOT
        DO 212 J=1,NCASE
C PLOT DATA WITH SYMBOLS
200     CALL MODE(4,HIGH,WIDE,9999.)
        CALL NOTE(XD(1,J),YD(1,J),ISYM(J),-NPTS(J))
```

```
      CALL MODE(4,.064,.042,9999.)
212   CONTINUE
C PUT ON TEXT
220   CALL MODE(1,1.,1.,0)
      CALL NOTE(0.0,-1.,IDINF1,72)
      CALL NOTE(0.0,-1.5,IDINF2,72)
C EXECUTE
      CALL DRAW(0.0,0.0,1,8000)
C SET UP: DISC BUFFER PLOT,1 COPY,PLOT IMMED.
      CALL MODE(0,2.0,1.0,1.0)
C END OF PLOT (CAUSES EXIT VIA VCOPY)
      CALL DRAW(0.0,0.0,0.0,9999)
C
901   FORMAT(8I8)
902   FORMAT('$NUMBER OF CASES [1 TO 5] >')
903   FORMAT(' INPUT X VALUE AND Y VALUE FOR CASE',I2)
904   FORMAT('$',I5,' >')
905   FORMAT(2E15.0)
906   FORMAT('$X AXIS MIN.,MAX. =',E12.4,E12.4)
908   FORMAT('$Y AXIS MIN.,MAX. ='E12.4,E12.4)
911   FORMAT(I5,2F10.4)
914   FORMAT(I5,8F10.4)
918   FORMAT(' INPUT AXIS LABEL FOR ',A1,' AXIS (15 CHARACTERS)')
919   FORMAT(6A4)
      END
C
C
C
      SUBROUTINE SIZSEL(ICHAR,HIGH,WIDE)
      DIMENSION JCHAR(8)
      DATA JCHAR/28,30,31,33,15,42,35,43/
      TYPE 100
      ACCEPT 103, ICHOIC
      TYPE 102,HIGH,WIDE
      ACCEPT 101, H, W
      IF(H.EQ.0..AND.W.EQ.0.) GO TO 10
      HIGH=H
      WIDE=W
10    ICHAR = JCHAR(ICHOIC)
      RETURN
100   FORMAT('   SELECT THE SYMBOL DESIRED:'//
      1 '    1= TRIANGLE'/
      2 '    2= CROSSED CIRCLE'/
      3 '    3= CROSSED SQUARE'/
      4 '    4= CROSSED DIAMOND'/
      5 '    5= CIRCLE'/
      6 '    6= ASTERISK'/
      7 '    7= POUND SIGN'/
      8 '    8= CROSS'/
      9 '$  SELECTION >')
101   FORMAT(2F10.5)
102   FORMAT(' TYPE THE DESIRED CHARACTER SIZE
      1 FOR SYMBOL PLOT.'/
      1 '$   HEIGHT & WIDTH ',F10.5,',',F10.5,' >')
```

```
103     FORMAT(I10)
        END
        SUBROUTINE MINMAX(RMIN,RMAX,RARRY,NCASE)
        DIMENSION RARRY(100,5)
        RMIN = 1E38
        RMAX = -1E38
        DO 10 I = 1,NCASE
        DO 20 J = 1, 100
        IF(RARRY(J,I).LT.RMIN.AND.RARRY(J,I).NE.0)RMIN=RARRY(J,I)
        IF(RARRY(J,I).GT.RMAX.AND.RARRY(J,I).NE.0)RMAX=RARRY(J,I)
20      CONTINUE
10      CONTINUE
        RETURN
        END

        SUBROUTINE GETNAM(FILNAM)
        BYTE FILNAM
        DIMENSION FILNAM(16)
        TYPE 100
        ACCEPT 101,(FILNAM(I),I=1,16)
        CALL NAMCLN(FILNAM)
        RETURN
100     FORMAT('$TYPE IN THE DEV:FILNAM.EXT FOR THE INPUT FILE>>')
101     FORMAT(16A1)
        END

        SUBROUTINE NAMCLN(FILNAM)
        BYTE FILNAM,DOT,SP
        DIMENSION FILNAM(16)
        DATA DOT/'.'/,SP/' '/
        I=1
2       IF (FILNAM(I).EQ.DOT)GO TO 1
        I=I+1
        GO TO 2
1       I=I+1
        DO 10 J = I, I+2
        IF(FILNAM(I).EQ.SP)FILNAM(J)="60
10      CONTINUE
        DO 11 J=I+3,16
        FILNAM(J)=0
11      CONTINUE
        RETURN
        END
```

The commands to build the program are:

```
FORTRA MIZPLT
R LINK
MIZPLT=MIZPLT,SYSLIB/F/C
MODE,NOTE,FORM,AXES,VPLIB,TCSLIB
```

VI. CLOSING COMMENTS

A few comments are in order regarding the future expandability of these programs, and the efficient use of these programs. A user as time progresses will in all likelihood develop "tool kits" of microprograms that are called into these programs to perform various functions. This tool building saves much time over the long term. In addition, since data files generated by both programs are identical in format, it is possible for data files from either program to be read by the lifetime program, plotted on a Tektronix 4014 terminal, and then to screen dump hard copies made for permanent reference. Figures 8, 9, and 10 are samples of such output. The format of A/I buffer files are also identical, so that the A/I data file plotting program is capable of plotting such data from either program. A sample of this program's output is shown in Figure 11. The design of the programs is well suited for expandability. In order to add a new operation, a new opcode is defined, an additional small selection and opcode insertion code segment placed into the programming mode, a line added to the program listing routine, and a new subroutine call added to the execution mode. The actual subroutine to execute the new operation is linked into the whole by adding to the linking command in the appendix. The hardware is also well suited for expansion. All that is necessary in order to add additional IEEE-488 devices to the system is an additional set of PDBs, ODBs, and a standard cable. For simple binary controlled devices, there are numerous additional lines available for both input and control available in the system. Drivers are written to mimic those contained in this report, and linked in with relative ease.

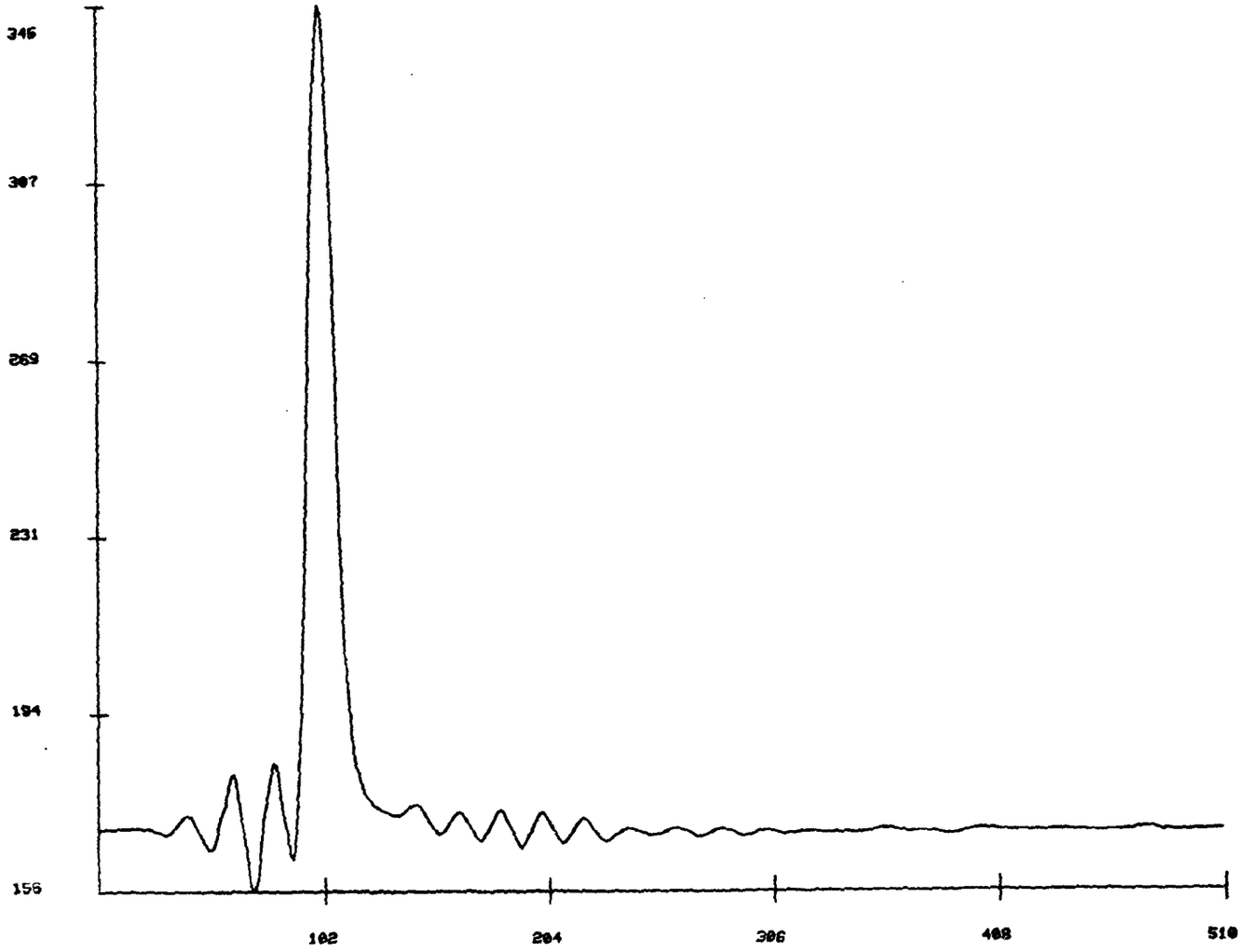


Figure 8. Single Buffer Data Plot Sample

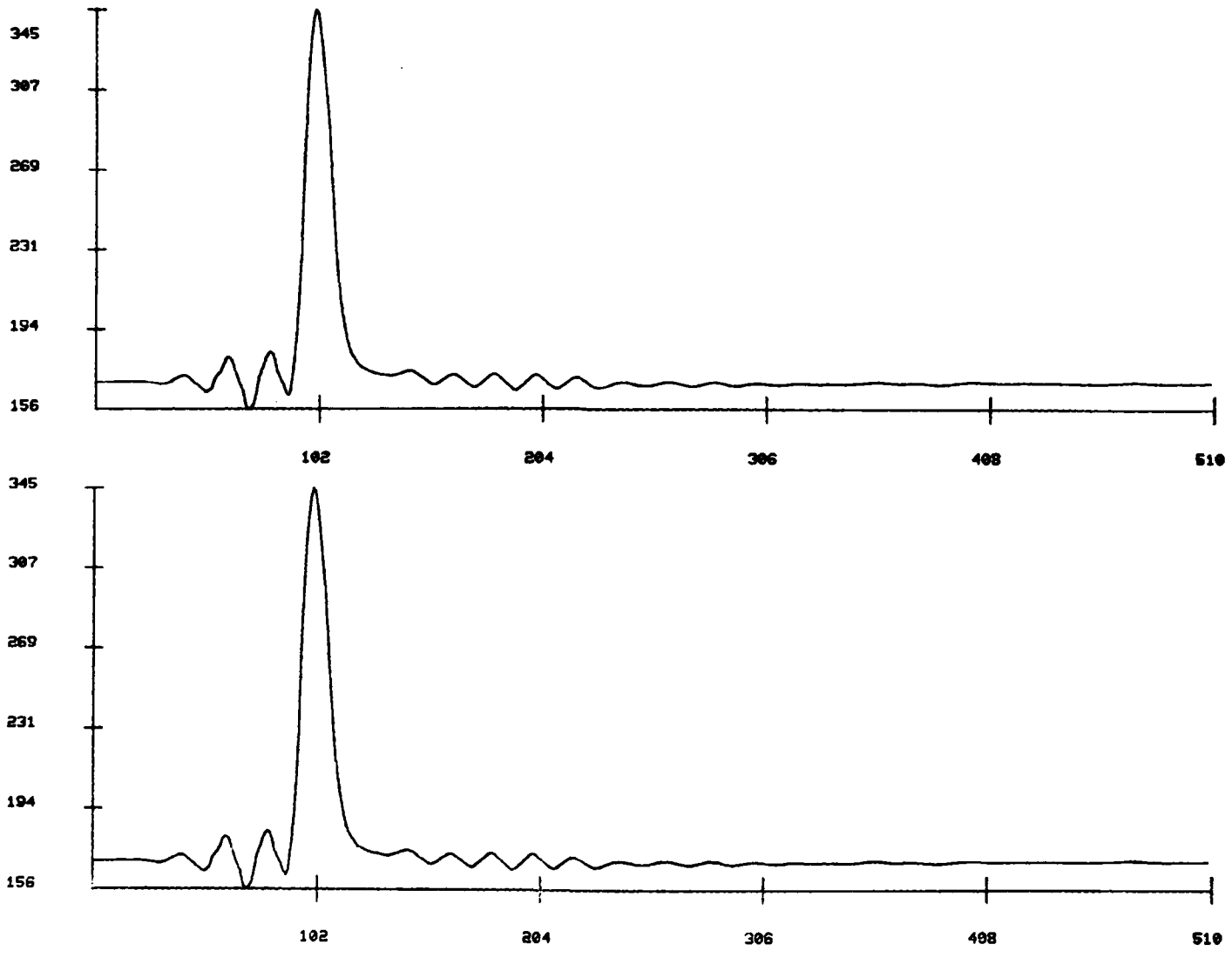


Figure 9. Double Buffer Data Plot Sample

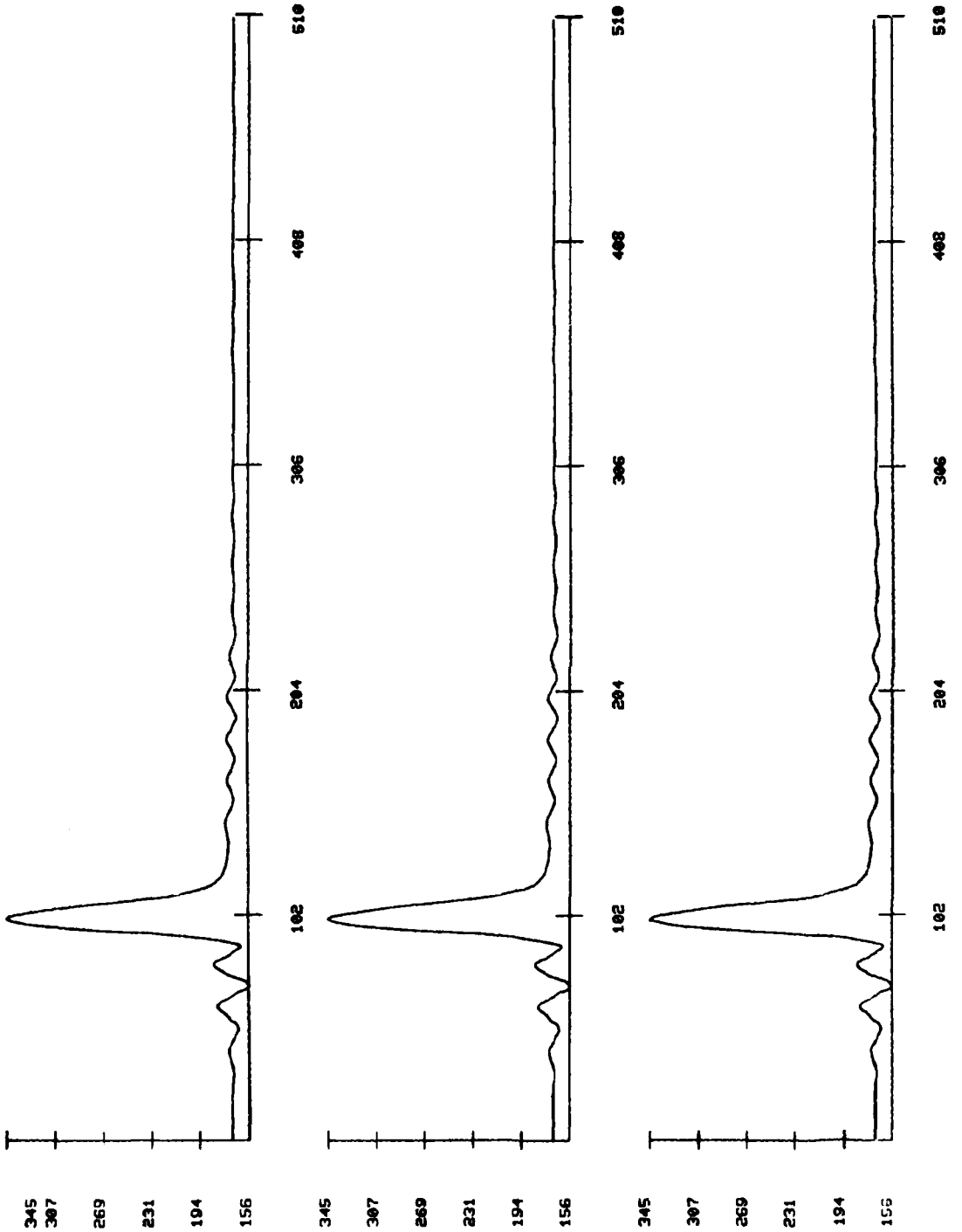


Figure 10. Triple Buffer Data Plot Sample

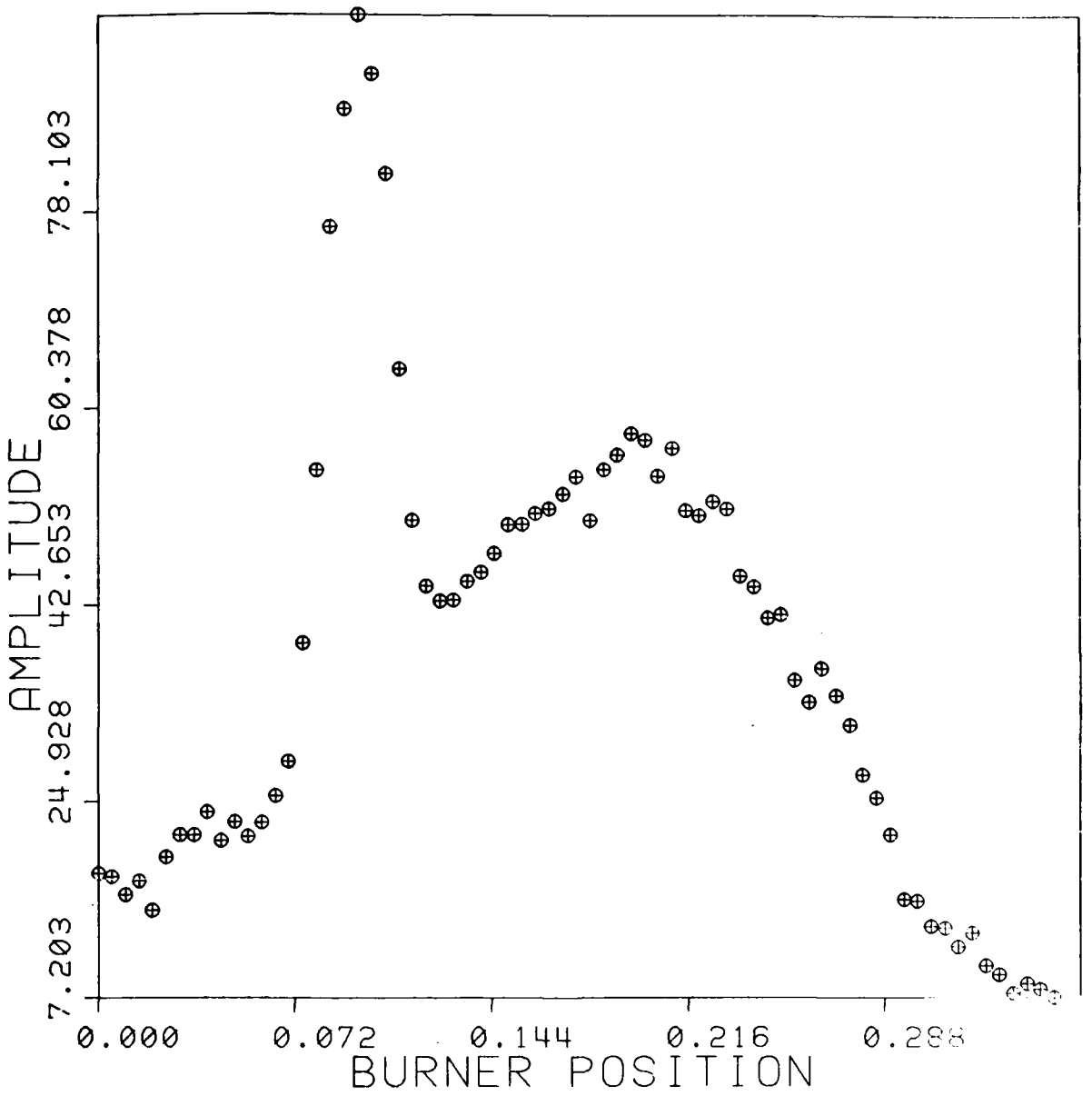


Figure 11.
A/I Buffer Plot from Auxiliary Plotting Program

VII. ACKNOWLEDGEMENTS

The authors wish to give our thanks and appreciation to our typist, Meg Griffith and Janet Krokowski for reformatting and preparing this rather lengthly report for publication.

REFERENCES

1. M.A. DeWilde, "The Combustion Diagnostics Laboratory Data Acquisition and Control System", BRL Technical report, in progress.

2. The 7912AD Transient Digitizer Operator's Manual, Part no. 070-2384-01, Tektronix, Inc., Beaverton, Oregon (1982).

3. M.A. DeWilde, "The Human Engineering of Data Acquisition and Control Systems", BRL-TR-2623, December, 1984.

4. Pulsed Dye Laser System User's Manual, Model PDL-1, Quanta-Ray, Inc., Mountain View, Calif. (1982).

5. Operational Design Manual for Versaplot Software, Part No. 50001 - 90001, Versatec, Inc., Santa Clara Calif., (1974).

APPENDIX

PROGRAM BUILDING COMMAND FILES

APPENDIX

PROGRAM BUILDING COMMAND FILES

A. The command for compiling the Lifetime program:

```
R FORTRA
OUP:LIFTIM.LFO,LST:LIFTIM=INP:LIFTIM.LIF/I:THR
OUP:CVTSCF.LFO,LST:CVTSCF=INP:CVTSCF.LIF/I:THR
OUP:NAMCLN.LFO,LST:NAMCLN=INP:NAMCLN.LIF/I:THR
OUP:EXECUT.LFO,LST:EXECUT=INP:EXECUT.LIF/I:THR
OUP:GETNAM.LFO,LST:GETNAM=INP:GETNAM.LIF/I:THR
OUP:OUTPRO.LFO,LST:OUTPRO=INP:OUTPRO.LIF/I:THR
OUP:GOTOMA.LFO,LST:GOTOMA=INP:GOTOMA.LIF/I:THR
OUP:PROGRM.LFO,LST:PROGRM=INP:PROGRM.LIF/I:THR
OUP:INPROG.LFO,LST:INPROG=INP:INPROG.LIF/I:THR
OUP:SETUP.LFO,LST:SETUP=INP:SETUP.LIF/I:THR
OUP:LISTPR.LFO,LST:LISTPR=INP:LISTPR.LIF/I:THR
OUP:ERABUF.LFO,LST:ERABUF=INP:ERABUF.LIF/I:THR
OUP:ACQSCF.LFO,LST:ACQSCF=INP:ACQSCF.LIF/I:THR
OUP:INCEXT.LFO,LST:INCEXT=INP:INCEXT.LIF/I:THR
OUP:ACUMMP.LFO,LST:ACUMMP=INP:ACUMMP.LIF/I:THR
OUP:ACUMSP.LFO,LST:ACUMSP=INP:ACUMSP.LIF/I:THR
OUP:INTGRT.LFO,LST:INTGRT=INP:INTGRT.LIF/I:THR
OUP:AIBIN.LFO,LST:AIBIN=INP:AIBIN.LIF/I:THR
OUP:AIBOUT.LFO,LST:AIBOUT=INP:AIBOUT.LIF/I:THR
OUP:AMPCUR.LFO,LST:AMPCUR=INP:AMPCUR.LIF/I:THR
OUP:PLOTIT.LFO,LST:PLOTIT=INP:PLOTIT.LIF/I:THR
OUP:AMPDAT.LFO,LST:AMPDAT=INP:AMPDAT.LIF/I:THR
OUP:MV2AI.LFO,LST:MV2AI=INP:MV2AI.LIF/I:THR
OUP:DATIN.LFO,LST:DATIN=INP:DATIN.LIF/I:THR
OUP:DATOUT.LFO,LST:DATOUT=INP:DATOUT.LIF/I:THR
OUP:DIFAB.LFO,LST:DIFAB=INP:DIFAB.LIF/I:THR
OUP:ERAAIB.LFO,LST:ERAAIB=INP:ERAAIB.LIF/I:THR
OUP:TYPAIB.LFO,LST:TYPAIB=INP:TYPAIB.LIF/I:THR
OUP:WFCR.LFO,LST:WFCR=INP:WFCR.LIF/I:THR
OUP:AMPLIT.LFO,LST:AMPLIT=INP:AMPLIT.LIF/I:THR
OUP:GETVAL.LFO,LST:GETVAL=INP:GETVAL.LIF/I:THR
OUP:CHNFIT.LFO,LST:CHNFIT=INP:CHNFIT.LIF/I:THR
OUP:CHNLIF.LFO,LST:CHNLIF=INP:CHNLIF.LIF/I:THR
OUP:STORE.LFO,LST:STORE=INP:STORE.LIF/I:THR
OUP:RESTOR.LFO,LST:RESTOR=INP:RESTOR.LIF/I:THR
OUP:AMPQND.LFO,LST:AMPQND=INP:AMPQND.LIF/I:THR
OUP:PLTAIB.LFO,LST:PLTAIB=INP:PLTAIB.LIF/I:THR
```

B. The command to link the LIFTIM program together:

```
R LINK
OUP:LIFTIM,MAP:LIFTIM=INP:LIFTIM.LFO,INP:CVTSCF.LFO/C
INP:NAMCLN.LFO,INP:EXECUT.LFO,INP:GETNAM.LFO/C
INP:GOTOMA.LFO,INP:PROGRM.LFO,INP:RESTOR.LFO/C
INP:FORLIB.SIM,INP:IEELIB,HSCALF,VSCALF/C
INP:DATA,INP:NUMSA,INP:PLTLIB/C
INP:ERABUF.LFO,INP:ACQSCF.LFO,INP:INCEXT.LFO/C
```

```
INP:AMPQND.LFO,INP:ACUMMP.LFO,INP:PLOTIT.LFO/C
INP:MV2AI.LFO,INP:DATIN.LFO,INP:DATOUT.LFO/C
INP:DIFAB.LFO,INP:ERAAIB.LFO,INP:PLTAIB.LFO/C
INP:TYPAIB.LFO,INP:WFCR.LFO,/C
INP:OUTPRO.LFO,INP:INPROG.LFO/O:1/C
INP:ACUMSP.LFO,INP:AIBIN.LFO,INP:AIBOUT.LFO/C
INP:INTGRT.LFO,INP:AMPLIT.LFO,INP:GETVAL.LFO/C
INP:AMPDAT.LFO,INP:AMPCUR.LFO/C
INP:LISTPR.LFO/O:1/C
INP:SETUP.LFO/O:1/C
INP:CHNFIT.LFO,INP:STORE.LFO
```

C. The command to generate a Lifetime program listing:

```
PRINT OPCODE.LIF,COMPIL.LIF,LINKER.LIF,CHAIN.LIF
PRINT LIFTIM.LIF,SETUP.LIF,PROGRM.LIF,LISTPR.LIF
PRINT OUTPRO.LIF,INPROG.LIF,STORE.LIF,RESTOR.LIF
PRINT EXECUT.LIF,ERABUF.LIF,ACUMSP.LIF,ACUMMP.LIF,ACQSCF.LIF
PRINT DATIN.LIF,DATOUT.LIF,INCEXT.LIF,GETNAM.LIF,NAMCLN.LIF
PRINT DIFAB.LIF,AMPDAT.LIF,AMPCUR.LIF,AMPQND.LIF,AMPLIT.LIF
PRINT INTGRT.LIF,WFCR.LIF,PLOTIT.LIF,CVTSCF.LIF,GETVAL.LIF
PRINT TYPAIB.LIF,ERAAIB.LIF,MV2AI.LIF,AIBIN.LIF,AIBOUT.LIF
PRINT CHNFIT.LIF,GOTOMA.LIF
PRINT CHNLIF.LIF,EXPFIT.LIF
PRINT FF
```

D. The command to compile the Multiphoton program:

```
R FORTRA
OUP:MULTIP.MZO,LST:MULTIP=INP:MULTIP.MIZ/I:THR
OUP:CVTSCF.MZO,LST:CVTSCF=INP:CVTSCF.MIZ/I:THR
OUP:NAMCLN.MZO,LST:NAMCLN=INP:NAMCLN.MIZ/I:THR
OUP:EXECUT.MZO,LST:EXECUT=INP:EXECUT.MIZ/I:THR
OUP:GETNAM.MZO,LST:GETNAM=INP:GETNAM.MIZ/I:THR
OUP:OUTPRO.MZO,LST:OUTPRO=INP:OUTPRO.MIZ/I:THR
OUP:GOTOMA.MZO,LST:GOTOMA=INP:GOTOMA.MIZ/I:THR
OUP:PROGRM.MZO,LST:PROGRM=INP:PROGRM.MIZ/I:THR
OUP:INPROG.MZO,LST:INPROG=INP:INPROG.MIZ/I:THR
OUP:SETUP.MZO,LST:SETUP=INP:SETUP.MIZ/I:THR
OUP:LISTPR.MZO,LST:LISTPR=INP:LISTPR.MIZ/I:THR
OUP:ERABUF.MZO,LST:ERABUF=INP:ERABUF.MIZ/I:THR
OUP:ACQSCF.MZO,LST:ACQSCF=INP:ACQSCF.MIZ/I:THR
OUP:INCEXT.MZO,LST:INCEXT=INP:INCEXT.MIZ/I:THR
OUP:ACUMMP.MZO,LST:ACUMMP=INP:ACUMMP.MIZ/I:THR
OUP:ACUMSP.MZO,LST:ACUMSP=INP:ACUMSP.MIZ/I:THR
OUP:INTGRT.MZO,LST:INTGRT=INP:INTGRT.MIZ/I:THR
OUP:AIBIN.MZO,LST:AIBIN=INP:AIBIN.MIZ/I:THR
OUP:AIBOUT.MZO,LST:AIBOUT=INP:AIBOUT.MIZ/I:THR
OUP:AMPCUR.MZO,LST:AMPCUR=INP:AMPCUR.MIZ/I:THR
OUP:MAKPLT.MZO,LST:MAKPLT=INP:MAKPLT.MIZ/I:THR
OUP:AMPDAT.MZO,LST:AMPDAT=INP:AMPDAT.MIZ/I:THR
OUP:MV2AI.MZO,LST:MV2AI=INP:MV2AI.MIZ/I:THR
OUP:DATIN.MZO,LST:DATIN=INP:DATIN.MIZ/I:THR
OUP:DATOUT.MZO,LST:DATOUT=INP:DATOUT.MIZ/I:THR
```

```
OUP:DIFAB.MZO,LST:DIFAB=INP:DIFAB.MIZ/I:THR
OUP:ERAAIB.MZO,LST:ERAAIB=INP:ERAAIB.MIZ/I:THR
OUP:TYPAIB.MZO,LST:TYPAIB=INP:TYPAIB.MIZ/I:THR
OUP:WFCR.MZO,LST:WFCR=INP:WFCR.MIZ/I:THR
OUP:AMPLIT.MZO,LST:AMPLIT=INP:AMPLIT.MIZ/I:THR
OUP:GETVAL.MZO,LST:GETVAL=INP:GETVAL.MIZ/I:THR
OUP:AMPQND.MZO,LST:AMPQND=INP:AMPQND.MIZ/I:THR
OUP:PLTAIB.MZO,LST:PLTAIB=INP:PLTAIB.MIZ/I:THR

R MACRO
OUP:MOVBUR.MZO,LST:MOVBUR=INP:MOVBUR.MIZ
OUP:SHUTTR.MZO,LST:SHUTTR=INP:SHUTTR.MIZ
OUP:JOLTS.MZO,LST:JOLTS=INP:JOLTS.MIZ
OUP:LASER.MZO,LST:LASER=INP:LASER.MIZ
```

E. The command to link the multiphoton program together:

```
R LINK
OUP:MULTIP,MAP:MULTIP=INP:MULTIP.MZO,INP:EXECUT.MZO,INP:MAKPLT.MZO/C
INP:DATA,INP:NUMSA,INP:GOSFOR,INP:PRTERR/C
INP:NAMCLN.MZO,INP:GETNAM.MZO/C
INP:FORLIB.SIM,INP:HSCALF,INP:VSCALF/C
INP:CVTSCF.MZO,INP:ACUMMP.MZO,INP:ACQSCF.MZO,INP:ACUMSP.MZO,INP:IEELIB/O:1/C
INP:PROGRM.MZO/O:1/C
INP:LISTPR.MZO/O:1/C
INP:SETUP.MZO/O:1/C
INP:AMPQND.MZO,INP:INTGRT.MZO,INP:AMPLIT.MZO/C
INP:GETVAL.MZO,INP:AMPDAT.MZO,INP:AMPCUR.MZO/C
INP:PLTAIB.MZO/O:1/C
INP:GOTOMA.MZO,INP:ERABUF.MZO,INP:INCEXT.MZO/C
INP:LASER.MZO,INP:MOVBUR.MZO/C
INP:MV2AI.MZO,INP:DATIN.MZO,INP:DATOUT.MZO/C
INP:DIFAB.MZO,INP:ERAAIB.MZO,INP:SHUTTR.MZO/C
INP:TYPAIB.MZO,INP:WFCR.MZO/C
INP:OUTPRO.MZO,INP:INPROG.MZO,INP:JOLTS.MZO/C
INP:AIBIN.MZO,INP:AIBOUT.MZO
```

F. The commands to build IEELIB, The IEEE-488 library:

```
R MACRO
INP:ENDIT,LST:ENDIT=INP:ENDIT.ODB
INP:RATC,LST:RATC=INP:RATC.ODB
INP:DIGDAT,LST:DIGDAT=INP:DIGDAT.ODB
INP:UNTL,LST:UNTL=INP:UNTL.ODB
INP:RBB,LST:RBB=INP:RBB.ODB
INP:MAKLSN,LST:MAKLSN=INP:MAKLSN.ODB
INP:MODDIG,LST:MODDIG=INP:MODDIG.ODB
INP:ATC,LST:ATC=INP:ATC.ODB
INP:MAKTLK,LST:MAKTLK=INP:MAKTLK.ODB
INP:LLO,LST:LLO=INP:LLO.ODB
INP:GTL,LST:GTL=INP:GTL.ODB
INP:RSCF,LST:RSCF=INP:RSCF.ODB
INP:MODTV,LST:MODTV=INP:MODTV.ODB
INP:GRATOF,LST:GRATOF=INP:GRATOF.ODB
```

```
INP:REASC1,LST:REASC1=INP:REASC1.ODB
INP:DUMP,LST:DUMP=INP:DUMP.PDB
INP:GETSCF,LST:GETSCF=INP:GETSCF.PDB
INP:DUMPSA,LST:DUMPSA=INP:DUMPSA.PDB
INP:DIGSA,LST:DIGSA=INP:DIGSA.ODB
INP:NUMSA,LST:NUMSA=INP:NUMSA
INP:DATA,LST:DATA=INP:DATA
INP:HSCALF,LST:HSCALF=INP:HSCALF
INP:VSCALF,LST:VSCALF=INP:VSCALF
INP:CIF,LST:CIF=CIF
INP:LSNR,LST:LSNR=LSNR
INP:TALKIT,LST:TALKIT=TALKIT
INP:OFFLIN,LST:OFFLIN=INP:OFFLIN.ODB
INP:NOWAIT,LST:NOWAIT=INP:NOWAIT.ODB
INP:REDSA,LST:REDSA=INP:REDSA.ODB
INP:SEMICO,LST:SEMICO=INP:SEMICO.ODB
INP:SEMIGO,LST:SEMIGO=INP:SEMIGO.ODB

R LIBR
INP:IEELIB,LST:IEELIB=INP:ENDIT,INP:RATC,INP:DIGDAT,INP:UNTL/C
INP:RBB,INP:MAKLSN,INP:MODDIG,INP:ATC,INP:MAKTLK,INP:LLO/C
INP:GTL,INP:RSCF,INP:MODTV,INP:GRATOF,INP:REASC1,INP:DUMP/C
INP:GETSCF,INP:DUMPSA,INP:DIGSA,INP:OFFLIN,INP:SEMIGO/C
INP:CIF,INP:TALKIT,INP:LSNR,INP:NOWAIT,INP:REDSA,INP:SEMICO

DELETE/NOQ  INP:ENDIT.OBJ,INP:RATC.OBJ,INP:DIGDAT.OBJ,INP:UNTL.OBJ
DELETE/NOQ  INP:RBB.OBJ,INP:MAKLSN.OBJ,INP:MODDIG.OBJ,INP:ATC.OBJ
DELETE/NOQ  INP:MAKTLK.OBJ,INP:LLO.OBJ,INP:GTL.OBJ,INP:RSCF.OBJ
DELETE/NOQ  INP:MODTV.OBJ,INP:GRATOF.OBJ,INP:REASC1.OBJ,INP:DUMP.OBJ
DELETE/NOQ  INP:GETSCF.OBJ,INP:DUMPSA.OBJ,INP:DIGSA.OBJ,INP:OFFLIN.OBJ
DELETE/NOQ  INP:CIF.OBJ,INP:LSNR.OBJ,INP:TALKIT.OBJ
DELETE/NOQ  INP:NOWAIT.OBJ,INP:REDSA.OBJ,INP:SEMICO.OBJ,INP:SEMIGO.OBJ
```

G. The linkage map for the Lifetime program:

```
RT-11 LINK  V06.01      Load Map
              Title:  LIFTIM  Ident:  FORV02


. ABS.   000000 001000   (RW,I,GBL,ABS,OVR)
                         $USRSW  000000  $RF2A1  000000  $HRDWR  000000
                         .VIR    000000  $NLCHN  000006  $SYSV$  000012
                         $WASIZ  000152  $LRECL  000210  $TRACE  004737
$OHAND   001000 000106   (RW,I,GBL,REL,CON)
                         $OVRH   001002  O$READ  001024  O$DONE  001036
                         $ODF1   001102  $ODF2   001104
$OTABL   001106 000152   (RW,D,GBL,REL,OVR)
OTS$I    001260 027674   (RW,I,LCL,REL,CON)
                         $$OTSI  001260  ADF$IM  001260  ADF$PM  001266
                         SUF$PM  001272  SUF$MM  001276  ADF$MM  001310
                         SUF$IM  001320  SUF$SM  001324  ADF$SM  001330
                         ADF$IP  001350  ADF$PP  001356  SUF$PP  001362
                         SUF$MP  001366  ADF$MP  001400  SUF$IP  001410
```

SUF$SP	001414	ADF$SP	001420	$CVTFB	001440
$CVTFI	001440	$CVTCB	001454	$CVTCI	001454
$CVTDB	001454	$CVTDI	001454	CIC$	001466
CID$	001466	CLC$	001466	CLD$	001466
$DI	001466	CIF$	001476	CLF$	001476
$RI	001476	CIL$	001610	CLI$	001614
$CVTIF	001616	$CVTIC	001632	$CVTID	001632
CCI$	001644	CDI$	001644	$IC	001644
$ID	001644	CFI$	001660	$IR	001660
RCI$	001744	GCO$	002714	FCO$	002722
ECO$	002726	DCO$	002734	ADF$IS	003656
ADF$PS	003664	SUF$PS	003670	SUF$MS	003674
ADF$MS	003706	SUF$IS	003716	$ADDF	003724
$SUBF	003740	SUF$SS	003752	$SBR	003752
ADF$SS	003756	$ADR	003756	ADD$	003772
DIF$PS	004416	DIF$MS	004422	DIF$IS	004432
$DIVF	004440	DIF$SS	004452	$DVR	004452
MUF$PS	004740	MUF$MS	004744	MUF$IS	004754
$MULF	004762	MUF$SS	004774	$MLR	004774
DII$PS	005304	DII$MS	005312	DII$IS	005316
DII$SS	005320	$DVI	005320	$OTI	005456
$$OTI	005460	$SETOP	005670	$$SET	007342
$XFI	007636	XFI$	007650	$PWRI	007650
$INITI	010126	ASSIGN	010244	IOR$	010670
AND$	010674	EQV$	010702	XOR$	010704
NMI$1M	010720	NMI$1I	010732	BLE$	010742
BEQ$	010744	BGT$	010752	BGE$	010754
BRA$	010756	BNE$	010762	BLT$	010764
CAI$	010774	CAL$	011002	$CLOSE	011032
CLOSE	011610	CMF$PS	011634	CMF$MS	011640
CMF$IS	011654	$CMPF	011662	CMF$SS	011674
$CMR	011674	CMF$PI	011706	CMF$MI	011712
CMF$II	011722	CMF$SI	011726	CMF$PP	011740
CMF$MP	011744	CMF$IP	011754	CMF$SP	011760
CMF$PM	011770	CMF$MM	011774	CMF$IM	012004
CMF$SM	012010	OCI$	012042	ICI$	012050
$ECI	012064	OCO$	012244	ICO$	012252
CPL$SM	012450	CPI$SM	012454	CPF$SM	012460
CPD$SM	012472	$DUMPL	012504	ENC$	012632
$ENC	012636	DEC$	012644	$DEC	012650
$OPNER	013034	$CHKER	013072	$IOEXI	013116
$EOL	013164	EOL$	013166	$ERRTB	013302
$ERRS	013407	EXIT	017150	$FCHNL	017154
$FIO	020016	$$FIO	020022	MOF$IS	021166
MOF$OS	021174	MOF$MS	021202	MOF$PS	021214
MOF$SM	021220	MOF$SP	021230	MOF$OM	021234
MOF$OA	021244	MOF$OP	021250	MOF$MM	021254
MOF$MA	021266	MOF$MP	021274	MOF$PM	021302
MOF$PA	021306	MOF$PP	021312	$GETFI	021316
$GETRE	021354	$TTYIN	021430	JMI$P	021564
JMI$M	021566	JMC$	021572	ADI$SS	021624
ADI$SA	021630	ADI$SM	021634	ADI$IS	021640
ADI$IA	021644	ADI$IM	021650	ADI$MS	021654
ADI$MA	021660	ADI$MM	021664	CMI$SS	021670

CMI$SI	021674	CMI$SM	021700	CMI$IS	021704
CMI$II	021710	CMI$IM	021714	CMI$MS	021720
CMI$MI	021724	CMI$MM	021730	IFR$	021734
$IFR	021740	$$IFR	021744	IFR$$	021776
IFW$	022020	$IFW	022024	$$IFW	022030
IFW$$	022066	ILW$	022136	$ILW	022142
TVS$	022262	$TVS	022264	MOI$SS	023076
MOL$SS	023076	MOI$SM	023102	MOI$SA	023106
MOI$IS	023112	MOL$IS	023112	REL$	023112
MOI$IM	023116	MOI$IA	023122	MOI$MS	023126
MOI$MM	023132	MOI$MA	023136	MOI$0S	023142
MOI$0M	023146	MOI$0A	023152	MOI$1S	023156
MOI$1M	023164	MOI$1A	023172	ICI$S	023200
ICI$M	023204	ICI$P	023210	ICI$A	023212
DCI$S	023216	DCI$M	023222	DCI$P	023226
DCI$A	023230	CMI$IP	023234	CMI$SP	023236
CMI$MP	023244	CMI$PP	023254	CMI$PS	023260
CMI$PI	023266	CMI$PM	023274	MOI$IP	023302
MOI$SP	023304	MOI$PP	023312	MOI$MP	023316
MOI$PS	023326	MOI$PM	023334	MOI$PA	023342
MOI$0P	023350	MOI$1P	023356	SUI$IP	023366
SUI$SP	023370	SUI$PP	023376	SUI$MP	023402
SUI$PS	023412	SUI$PM	023420	SUI$PA	023426
ISN$	023434	$ISNTR	023440	LSN$	023454
$LSNTR	023460	SUI$SS	023614	SUI$SA	023620
SUI$SM	023624	SUI$IS	023630	SUI$IA	023634
SUI$IM	023640	SUI$MS	023644	SUI$MA	023650
SUI$MM	023654	CML$IS	023660	CML$SS	023666
CML$IM	023674	CML$SM	023676	CML$MS	023702
CML$MM	023706	MOL$SM	023712	MOL$SA	023716
MOL$MS	023722	MOL$MM	023732	MOL$MA	023736
MOL$SP	023742	MOL$PP	023750	MOL$MP	023754
MOL$PM	023764	MOL$PS	023772	MOL$PA	023776
MOL$IM	024004	MOL$IA	024012	MOL$IP	024020
LLE$	024030	LEQ$	024032	LGT$	024040
LGE$	024042	LNE$	024052	LLT$	024054
CML$PS	024060	CML$PI	024066	CML$PM	024076
TSL$S	024104	TSL$M	024110	TSL$I	024114
TSL$P	024122	NMI$II	024130	NMI$MI	024166
NMI$PI	024174	NPI$II	024204	NPI$MI	024210
NPI$PI	024214	$$OPCL	024220	$$$ERR	024332
$$$DIS	024354	$OSTMI	024456	$OSTM	024462
$PSE	025612	$PSES	025646	BAH$	025676
PSE$	025712	$PUTRE	025756	RET$L	026264
RET$F	026270	RET$I	026276	RET$	026300
IRR$	026334	$IRR	026340	IRW$	026364
$IRW	026370	$GETIN	026744	$SETIN	027002
DEF$	027110	$DEF	027114	$PUTBL	027210
$GETBL	027420	$EOFIL	027604	$EOF2	027620
SAVRG$	027640	THRD$	030016	SAVR4$	030020
$STPS	030104	STP$	030112	$STP	030112
FOO$	030116	$EXIT	030136	$OTIS	030262
$$OTIS	030264	TSI$S	030404	TSD$S	030410
TSF$S	030414	TSD$M	030420	TSF$M	030420

```
                        TSI$M   030420  TSD$I   030424  TSF$I   030424
                        TSI$I   030424  TSD$P   030430  TSF$P   030430
                        TSI$P   030430  TVL$    030436  $TVL    030436
                        TVF$    030444  $TVF    030444  TVD$    030452
                        $TVD    030452  TVQ$    030460  $TVQ    030460
                        TVP$    030466  $TVP    030466  TVI$    030474
                        $TVI    030474  $WAIT   030630  $VRINT  030672
                        SAF$IM  030704  SAF$SM  030706  SVF$IM  030720
                        SVF$SM  030722  SAF$MM  030742  SVF$MM  030746
                        SAF$IP  030752  SAF$SP  030754  SVF$IP  030766
                        SVF$SP  030770  SAF$MP  031010  SVF$MP  031014
                        SAI$IM  031020  SAI$SM  031022  $BOUND  031026
                        SVI$IM  031052  SVI$SM  031054  SAI$MM  031064
                        SVI$MM  031070  SAL$IM  031074  SAL$SM  031076
                        SVL$IM  031104  SVL$SM  031106  SAL$MM  031114
                        SVL$MM  031120  SAL$IP  031124  SAL$SP  031126
                        SVL$IP  031134  SVL$SP  031136  SAL$MP  031144
                        SVL$MP  031150
OTS$P   031154 000054    (RW,D,GBL,REL,OVR)
SYS$I   031230 000042    (RW,I,LCL,REL,CON)
                        RCHAIN  031230
USER$I  031272 000000    (RW,I,LCL,REL,CON)
$CODE   031272 023130    (RW,I,LCL,REL,CON)
                        $$OTSC  031272  CVTSCF  032242  SORTIT  032320
                        NAMCLN  033012  EXECUT  033314  GETNAM  035364
                        GOTOMA  035514  PROGRM  035610  GETBUF  040474
                        GETFIL  040720  RESTOR  041052  ERABUF  042130
                        ACQSCF  042414  INCEXT  042512  INCR    042576
                        AMPQND  043202  BASLIN  043470  ACUMMP  043604
                        SUMIT   044524  MAKPLT  045030  PICTUR  045554
                        AXES    046044  SCANER  046540  PLOTER  046720
                        LABEL   047442  INITT   050312  MV2AI   050432
                        DATIN   050470  INP     051200  DATOUT  051662
                        OUP     052372  DIFAB   053054  ERAAIB  053212
                        PLTAIB  053306  TYPAIB  054204  WFCR    054344
OTS$O   054422 001036    (RW,I,LCL,REL,CON)
                        $$OTSO  054422  $OPEN   054422
SYS$O   055460 000000    (RW,I,LCL,REL,CON)
$DATAP  055460 005456    (RW,D,LCL,REL,CON)
OTS$D   063136 000052    (RW,D,LCL,REL,CON)
                        NHCLN$  063142
OTS$S   063210 000004    (RW,D,LCL,REL,CON)
                        $AOTS   063210
SYS$S   063214 000004    (RW,D,LCL,REL,CON)
                        $SYSLB  063214  $LOCK   063216  $CRASH  063217
$DATA   063220 000712    (RW,D,LCL,REL,CON)
USER$D  064132 000000    (RW,D,LCL,REL,CON)
.$$$$.  064132 000000    (RW,D,GBL,REL,OVR)
DATA    064132 002000    (RW,D,GBL,REL,OVR)
                        DATA    064132
VSCALF  066132 000024    (RW,D,GBL,REL,OVR)
                        VSCALF  066132
HSCALF  066156 000024    (RW,D,GBL,REL,OVR)
                        HSCALF  066156
```

```
NUMSA   066202 000004   (RW,D,GBL,REL,OVR)
                       NUMSA   066202
BUFFA   066206 004000   (RW,D,GBL,REL,OVR)
BUFFB   072206 004000   (RW,D,GBL,REL,OVR)
BUFFD   076206 004000   (RW,D,GBL,REL,OVR)
CNTRL   102206 000426   (RW,D,GBL,REL,OVR)
ID      102634 000220   (RW,D,GBL,REL,OVR)
MUPROG  103054 000312   (RW,D,GBL,REL,OVR)
AI      103366 000632   (RW,D,GBL,REL,OVR)
PLT     104220 000014   (RW,D,GBL,REL,OVR)
CURSOR  104234 000022   (RW,D,GBL,REL,OVR)
PLTLIM  104256 000004   (RW,D,GBL,REL,OVR)
        104262 003744   (RW,I,LCL,REL,CON)
                       ENDIT   104262  RATC    104270  DIGDAT  104304
                       UNTL    104316  RBB     104326  MAKLSN  104420
                       MODDIG  104430  MAKTLK  104442  LLO     104452
                       GTL     104460  RSCF    104466  REASCl  104516
                       GETSCF  104532  DUMPSA  104560  DIGSA   104620
                       OFFLIN  104632  SEMIGO  104642  CIF     104652
                       BYTCNT  105754  CKSUMR  105756  CKSUMF  105760
                       OUTWRD  105770  DWA     105772  ENDDEL  106004
                       EOIFLG  106006  WAITFL  106010  NOWAIT  106012
                       REDSA   106016  DUMP    106032  ATC     106066
                       TALKIT  106074  LSNR    106416  SETTRM  107150
                       CSR     107176  BELL    107202  LINOUT  107236
                       ALFMOD  107302  MOVE    107336  DRAW    107346
                       TRMOUT  107626  TKMODE  107672  VTMODE  107736
                       DELAY   110036  VTPAGE  110056  ERASE   110156
Segment size = 110226 = 18507. words


OTS$I   110230 000150   (RW,I,LCL,REL,CON)
                       MUI$PS  110230  MUI$MS  110236  MUI$IS  110242
                       MUI$SS  110244  $MLI    110244  ABS     110324
                       MOF$RS  110342  MOF$RM  110350  MOF$RA  110360
                       MOF$RP  110364  CML$MI  110370  CML$SI  110372
SYS$I   110400 000000   (RW,I,LCL,REL,CON)
USER$I  110400 000000   (RW,I,LCL,REL,CON)
$CODE   110400 007532   (RW,I,LCL,REL,CON)
                       OUTPRO@ 110400  INPROG@ 110602  ACUMSP@ 111004
                       AIBIN @ 111340  AIBOUT@ 112056  INTGRT@ 112526
                       TRAP    113134  AMPLIT  113566  GETVAL  114056
                       AMPDAT@ 114724  AMPCUR@ 115074
OTS$O   115132 000000   (RW,I,LCL,REL,CON)
SYS$O   115132 000000   (RW,I,LCL,REL,CON)
$DATAP  115132 000450   (RW,D,LCL,REL,CON)
OTS$D   115602 000000   (RW,D,LCL,REL,CON)
OTS$S   115602 000000   (RW,D,LCL,REL,CON)
SYS$S   115602 000000   (RW,D,LCL,REL,CON)
$DATA   115602 000356   (RW,D,LCL,REL,CON)
USER$D  116160 000000   (RW,D,LCL,REL,CON)
        116160 000230   (RW,I,LCL,REL,CON)
                       GRAFIN  116160
Segment size = 006160 = 1592.  words
```

```
OTS$I    110230 000000   (RW,I,LCL,REL,CON)
SYS$I    110230 000000   (RW,I,LCL,REL,CON)
USER$I   110230 000000   (RW,I,LCL,REL,CON)
$CODE    110230 003624   (RW,I,LCL,REL,CON)
                        LISTPR@ 110230
OTS$O    114054 000000   (RW,I,LCL,REL,CON)
SYS$O    114054 000000   (RW,I,LCL,REL,CON)
$DATAP   114054 003632   (RW,D,LCL,REL,CON)
OTS$D    117706 000000   (RW,D,LCL,REL,CON)
OTS$S    117706 000000   (RW,D,LCL,REL,CON)
SYS$S    117706 000000   (RW,D,LCL,REL,CON)
$DATA    117706 000020   (RW,D,LCL,REL,CON)
USER$D   117726 000000   (RW,D,LCL,REL,CON)
Segment size = 007476 = 1951.  words


OTS$I    110230 000000   (RW,I,LCL,REL,CON)
SYS$I    110230 000042   (RW,I,LCL,REL,CON)
                        CHAIN   110230
USER$I   110272 000000   (RW,I,LCL,REL,CON)
$CODE    110272 002772   (RW,I,LCL,REL,CON)
                        SETUP @ 110272  CHNFIT@ 112126  STORE   112206
OTS$O    113264 000000   (RW,I,LCL,REL,CON)
SYS$O    113264 000000   (RW,I,LCL,REL,CON)
$DATAP   113264 002202   (RW,D,LCL,REL,CON)
OTS$D    115466 000000   (RW,D,LCL,REL,CON)
OTS$S    115466 000000   (RW,D,LCL,REL,CON)
SYS$S    115466 000000   (RW,D,LCL,REL,CON)
$DATA    115466 000104   (RW,D,LCL,REL,CON)
USER$D   115572 000000   (RW,D,LCL,REL,CON)
Segment size = 005342 = 1393.  words
```

The linkage map for the Multiphoton program:

```
RT-11 LINK  V06.01      Load Map        Tue 10-Jan-84 01:47:32
MULTIP.SAV      Title:  MULTIP  Ident:  FORV02


. ABS.   000000 001000   (RW,I,GBL,ABS,OVR)
                        $USRSW  000000  $RF2A1  000000  $HRDWR  000000
                        .VIR    000000  $NLCHN  000006  $SYSV$  000012
                        $WASIZ  000152  $LRECL  000210  $TRACE  004737
 $OHAND  001000 000106   (RW,I,GBL,REL,CON)
                        $OVRH   001002  O$READ  001024  O$DONE  001036
                        $ODF1   001102  $ODF2   001104
 $OTABL  001106 000406   (RW,D,GBL,REL,OVR)
 OTS$I   001514 024204   (RW,I,LCL,REL,CON)
                        $$OTSI  001514  ADF$IM  001514  ADF$PM  001522
```

SUF$PM	001526	SUF$MM	001532	ADF$MM	001544
SUF$IM	001554	SUF$SM	001560	ADF$SM	001564
ADF$IP	001604	ADF$PP	001612	SUF$PP	001616
SUF$MP	001622	ADF$MP	001634	SUF$IP	001644
SUF$SP	001650	ADF$SP	001654	$CVTFB	001674
$CVTFI	001674	$CVTCB	001710	$CVTCI	001710
$CVTDB	001710	$CVTDI	001710	CIC$	001722
CID$	001722	CLC$	001722	CLD$	001722
$DI	001722	CIF$	001732	CLF$	001732
$RI	001732	CIL$	002044	CLI$	002050
$CVTIF	002052	$CVTIC	002066	$CVTID	002066
CCI$	002100	CDI$	002100	$IC	002100
$ID	002100	CFI$	002114	$IR	002114
RCI$	002200	GCO$	003150	FCO$	003156
ECO$	003162	DCO$	003170	ADF$IS	004112
ADF$PS	004120	SUF$PS	004124	SUF$MS	004130
ADF$MS	004142	SUF$IS	004152	$ADDF	004160
$SUBF	004174	SUF$SS	004206	$SBR	004206
ADF$SS	004212	$ADR	004212	ADD$	004226
DIF$PS	004652	DIF$MS	004656	DIF$IS	004666
$DIVF	004674	DIF$SS	004706	$DVR	004706
MUF$PS	005174	MUF$MS	005200	MUF$IS	005210
$MULF	005216	MUF$SS	005230	$MLR	005230
MUI$PS	005540	MUI$MS	005546	MUI$IS	005552
MUI$SS	005554	$MLI	005554	$OTI	005662
$$OTI	005664	$SETOP	006074	$$SET	007546
$INITI	010042	IOR$	010160	AND$	010164
EQV$	010172	XOR$	010174	NMI$1M	010210
NMI$1I	010222	BLE$	010232	BEQ$	010234
BGT$	010242	BGE$	010244	BRA$	010246
BNE$	010252	BLT$	010254	CAI$	010264
CAL$	010272	$CLOSE	010322	CMF$PS	011100
CMF$MS	011104	CMF$IS	011120	$CMPF	011126
CMF$SS	011140	$CMR	011140	CMF$PI	011152
CMF$MI	011156	CMF$II	011166	CMF$SI	011172
CMF$PP	011204	CMF$MP	011210	CMF$IP	011220
CMF$SP	011224	CMF$PM	011234	CMF$MM	011240
CMF$IM	011250	CMF$SM	011254	OCI$	011306
ICI$	011314	$ECI	011330	OCO$	011510
ICO$	011516	CPL$SM	011714	CPI$SM	011720
CPF$SM	011724	CPD$SM	011736	$DUMPL	011750
ENC$	012076	$ENC	012102	DEC$	012110
$DEC	012114	$OPNER	012300	$CHKER	012336
$IOEXI	012362	$EOL	012430	EOL$	012432
$ERRTB	012546	$ERRS	012653	EXIT	016414
$FCHNL	016420	$FIO	017262	$$FIO	017266
MOF$IS	020432	MOF$OS	020440	MOF$MS	020446
MOF$PS	020460	MOF$SM	020464	MOF$SP	020474
MOF$OM	020500	MOF$OA	020510	MOF$OP	020514
MOF$MM	020520	MOF$MA	020532	MOF$MP	020540
MOF$PM	020546	MOF$PA	020552	MOF$PP	020556
$GETRE	020562	$TTYIN	020636	JMI$P	020772
JMI$M	020774	JMC$	021000	ADI$SS	021032
ADI$SA	021036	ADI$SM	021042	ADI$IS	021046

ADI$IA	021052	ADI$IM	021056	ADI$MS	021062
ADI$MA	021066	ADI$MM	021072	CMI$SS	021076
CMI$SI	021102	CMI$SM	021106	CMI$IS	021112
CMI$II	021116	CMI$IM	021122	CMI$MS	021126
CMI$MI	021132	CMI$MM	021136	IFR$	021142
$IFR	021146	$$IFR	021152	IFR$$	021204
IFW$	021226	$IFW	021232	$$IFW	021236
IFW$$	021274	ILW$	021344	$ILW	021350
TVS$	021470	$TVS	021472	MOI$SS	022304
MOL$SS	022304	MOI$SM	022310	MOI$SA	022314
MOI$IS	022320	MOL$IS	022320	REL$	022320
MOI$IM	022324	MOI$IA	022330	MOI$MS	022334
MOI$MM	022340	MOI$MA	022344	MOI$0S	022350
MOI$0M	022354	MOI$0A	022360	MOI$1S	022364
MOI$1M	022372	MOI$1A	022400	ICI$S	022406
ICI$M	022412	ICI$P	022416	ICI$A	022420
DCI$S	022424	DCI$M	022430	DCI$P	022434
DCI$A	022436	CMI$IP	022442	CMI$SP	022444
CMI$MP	022452	CMI$PP	022462	CMI$PS	022466
CMI$PI	022474	CMI$PM	022502	MOI$IP	022510
MOI$SP	022512	MOI$PP	022520	MOI$MP	022524
MOI$PS	022534	MOI$PM	022542	MOI$PA	022550
MOI$0P	022556	MOI$1P	022564	SUI$IP	022574
SUI$SP	022576	SUI$PP	022604	SUI$MP	022610
SUI$PS	022620	SUI$PM	022626	SUI$PA	022634
ISN$	022642	$ISNTR	022646	LSN$	022662
$LSNTR	022666	SUI$SS	023022	SUI$SA	023026
SUI$SM	023032	SUI$IS	023036	SUI$IA	023042
SUI$IM	023046	SUI$MS	023052	SUI$MA	023056
SUI$MM	023062	CML$IS	023066	CML$SS	023074
CML$IM	023102	CML$SM	023104	CML$MS	023110
CML$MM	023114	MOL$SM	023120	MOL$SA	023124
MOL$MS	023130	MOL$MM	023140	MOL$MA	023144
MOL$SP	023150	MOL$PP	023156	MOL$MP	023162
MOL$PM	023172	MOL$PS	023200	MOL$PA	023204
MOL$IM	023212	MOL$IA	023220	MOL$IP	023226
LLE$	023236	LEQ$	023240	LGT$	023246
LGE$	023250	LNE$	023260	LLT$	023262
CML$PS	023266	CML$PI	023274	CML$PM	023304
TSL$S	023312	TSL$M	023316	TSL$I	023322
TSL$P	023330	NMI$II	023336	NMI$MI	023374
NMI$PI	023402	NPI$II	023412	NPI$MI	023416
NPI$PI	023422	$PUTRE	023426	RET$L	023734
RET$F	023740	RET$I	023746	RET$	023750
$PUTBL	024004	$GETBL	024214	$EOFIL	024400
$EOF2	024414	SAVRG$	024434	THRD$	024612
$STPS	024614	STP$	024622	$STP	024622
FOO$	024626	$EXIT	024646	$OTIS	024772
$$OTIS	024774	TSI$S	025114	TSD$S	025120
TSF$S	025124	TSD$M	025130	TSF$M	025130
TSI$M	025130	TSD$I	025134	TSF$I	025134
TSI$I	025134	TSD$P	025140	TSF$P	025140
TSI$P	025140	TVL$	025146	$TVL	025146
TVF$	025154	$TVF	025154	TVD$	025162

```
                        $TVD    025162  TVQ$    025170  $TVQ    025170
                        TVP$    025176  $TVP    025176  TVI$    025204
                        $TVI    025204  $WAIT   025340  $VRINT  025402
                        SAF$IM  025414  SAF$SM  025416  SVF$IM  025430
                        SVF$SM  025432  SAF$MM  025452  SVF$MM  025456
                        SAF$IP  025462  SAF$SP  025464  SVF$IP  025476
                        SVF$SP  025500  SAF$MP  025520  SVF$MP  025524
                        SAI$IM  025530  SAI$SM  025532  $BOUND  025536
                        SVI$IM  025562  SVI$SM  025564  SAI$MM  025574
                        SVI$MM  025600  SAI$IP  025604  SAI$SP  025606
                        SVI$IP  025616  SVI$SP  025620  SAI$MP  025630
                        SVI$MP  025634  SAL$IM  025640  SAL$SM  025642
                        SVL$IM  025650  SVL$SM  025652  SAL$MM  025660
                        SVL$MM  025664  SAL$IP  025670  SAL$SP  025672
                        SVL$IP  025700  SVL$SP  025702  SAL$MP  025710
                        SVL$MP  025714
OTS$P   025720  000054   (RW,D,GBL,REL,OVR)
SYS$I   025774  000000   (RW,I,LCL,REL,CON)
USER$I  025774  000000   (RW,I,LCL,REL,CON)
$CODE   025774  010574   (RW,I,LCL,REL,CON)
                        $$OTSC  025774  EXECUT  026574  MAKPLT  030636
                        PICTUR  031412  AXES    031736  SCANER  032400
                        PLOTER  032560  LABMDP  033224  SETMDP  034666
                        DRWMDP  035156  ERATXT  035452  TYPMDP  035630
                        PRTERR  036010  NAMCLN  036136  GETNAM  036440
OTS$O   036570  001036   (RW,I,LCL,REL,CON)
                        $$OTSO  036570  $OPEN   036570
SYS$O   037626  000000   (RW,I,LCL,REL,CON)
$DATAP  037626  001420   (RW,D,LCL,REL,CON)
OTS$D   041246  000006   (RW,D,LCL,REL,CON)
                        NHCLN$  041252
OTS$S   041254  000002   (RW,D,LCL,REL,CON)
                        $AOTS   041254
SYS$S   041256  000000   (RW,D,LCL,REL,CON)
$DATA   041256  001720   (RW,D,LCL,REL,CON)
USER$D  043176  000000   (RW,D,LCL,REL,CON)
.$$$$.  043176  000000   (RW,D,GBL,REL,OVR)
DATA    043176  002000   (RW,D,GBL,REL,OVR)
                        DATA    043176
VSCALF  045176  000024   (RW,D,GBL,REL,OVR)
                        VSCALF  045176
HSCALF  045222  000024   (RW,D,GBL,REL,OVR)
                        HSCALF  045222
NUMSA   045246  000004   (RW,D,GBL,REL,OVR)
                        NUMSA   045246
BUFFA   045252  004000   (RW,D,GBL,REL,OVR)
BUFFB   051252  004000   (RW,D,GBL,REL,OVR)
BUFFD   055252  004000   (RW,D,GBL,REL,OVR)
CNTRL   061252  000424   (RW,D,GBL,REL,OVR)
ID      061676  000220   (RW,D,GBL,REL,OVR)
MUPROG  062116  000310   (RW,D,GBL,REL,OVR)
AI      062426  000632   (RW,D,GBL,REL,OVR)
PLT     063260  000014   (RW,D,GBL,REL,OVR)
CURSOR  063274  000022   (RW,D,GBL,REL,OVR)
```

```
PLTLIM  063316 000004   (RW,D,GBL,REL,OVR)
GOSFOR  063322 003506   (RW,I,LCL,REL,CON)
                        INITGA  063322  STARTG  063454  CLRMDP  063520
                        WFMDP   063536  PHYSAD  063572  DLINE   063612
                        OLINE   063636  CIRCLE  063664  ELLIPS  063712
                        DPOINT  063740  OPOINT  063762  TEXT    064004
                        ITEXT   064022  DFILL   064040  OFILL   064066
                        DCLEAR  064114  OCLEAR  064130  COLOR   064144
                        COLT    064164  DMA     064202  RGM     064222
                        WGM     064246  XRMDP   064272  WMDP    064312
                        RMDP    064332  SETCLR  064352  SETMOD  064370
                        SETOVR  064406  GOSIN   064424  DISPLA  064440
                        SETZOM  064456  PICMOV  064500  GRID    064522
                        WINDOW  064540  MOVIE   064560  CURSOR  064576
                        CRSOFF  064620  WFTRM   064634  DRAGON  064650
                        DRGOFF  064666  CHECK1  064702  CHECK2  064716
                        USRPGM  064732  DELAY   065000  XNOP    065016
                        SETTXT  065032  CLRTXT  065046  SETGR   065062
                        CLRGR   065076  SETOVD  065112  CLROVD  065130
                        WAITVR  065146  RCALL   065162  SETEXT  065200
                        CLREXT  065214  JUMP    065230  XGOS    066744
MAPRO   067030 000010   (RW,I,LCL,REL,CON)
                        MAPRO   067030
Segment size = 067040 = 14096. words


OTS$I   067042 000270   (RW,I,LCL,REL,CON)
                        $XFI    067042  XFI$    067054  $PWRI   067054
SYS$I   067332 000000   (RW,I,LCL,REL,CON)
USER$I  067332 000000   (RW,I,LCL,REL,CON)
$CODE   067332 002426   (RW,I,LCL,REL,CON)
                        CVTSCF  067332  SORTIT  067410  ACUMMP@ 070102
                        SUMIT   071022  ACQSCF@ 071326  ACUMSP@ 071424
OTS$O   071760 000000   (RW,I,LCL,REL,CON)
SYS$O   071760 000000   (RW,I,LCL,REL,CON)
$DATAP  071760 000222   (RW,D,LCL,REL,CON)
OTS$D   072202 000000   (RW,D,LCL,REL,CON)
OTS$S   072202 000000   (RW,D,LCL,REL,CON)
SYS$S   072202 000000   (RW,D,LCL,REL,CON)
$DATA   072202 000122   (RW,D,LCL,REL,CON)
USER$D  072324 000000   (RW,D,LCL,REL,CON)
        072324 002666   (RW,I,LCL,REL,CON)
                        ENDIT   072324  RATC    072332  DIGDAT  072346
                        UNTL    072360  RBB     072370  MAKLSN  072462
                        MODDIG  072472  MAKTLK  072504  LLO     072514
                        GTL     072522  RSCF    072530  REASC1  072560
                        GETSCF  072574  DUMPSA  072622  DIGSA   072662
                        OFFLIN  072674  SEMIGO  072704  CIF     072714
                        BYTCNT  074016  CKSUMR  074020  CKSUMF  074022
                        OUTWRD  074032  DWA     074034  ENDDEL  074046
                        EOIFLG  074050  WAITFL  074052  NOWAIT  074054
                        REDSA   074060  DUMP    074074  ATC     074130
                        TALKIT  074136  LSNR    074460
Segment size = 006150 = 1588.  words
```

```
OTS$I   067042 000000   (RW,I,LCL,REL,CON)
SYS$I   067042 000000   (RW,I,LCL,REL,CON)
USER$I  067042 000000   (RW,I,LCL,REL,CON)
$CODE   067042 004016   (RW,I,LCL,REL,CON)
                       PROGRM@ 067042  GETBUF  072502  GETFIL  072726
OTS$O   073060 000000   (RW,I,LCL,REL,CON)
SYS$O   073060 000000   (RW,I,LCL,REL,CON)
$DATAP  073060 004136   (RW,D,LCL,REL,CON)
OTS$D   077216 000000   (RW,D,LCL,REL,CON)
OTS$S   077216 000000   (RW,D,LCL,REL,CON)
SYS$S   077216 000000   (RW,D,LCL,REL,CON)
$DATA   077216 000050   (RW,D,LCL,REL,CON)
USER$D  077266 000000   (RW,D,LCL,REL,CON)
Segment size = 010224 = 2122.  words


OTS$I   067042 000000   (RW,I,LCL,REL,CON)
SYS$I   067042 000000   (RW,I,LCL,REL,CON)
USER$I  067042 000000   (RW,I,LCL,REL,CON)
$CODE   067042 004100   (RW,I,LCL,REL,CON)
                       LISTPR@ 067042
OTS$O   073142 000000   (RW,I,LCL,REL,CON)
SYS$O   073142 000000   (RW,I,LCL,REL,CON)
$DATAP  073142 004044   (RW,D,LCL,REL,CON)
OTS$D   077206 000000   (RW,D,LCL,REL,CON)
OTS$S   077206 000000   (RW,D,LCL,REL,CON)
SYS$S   077206 000000   (RW,D,LCL,REL,CON)
$DATA   077206 000030   (RW,D,LCL,REL,CON)
USER$D  077236 000000   (RW,D,LCL,REL,CON)
Segment size = 010174 = 2110.  words


OTS$I   067042 001664   (RW,I,LCL,REL,CON)
                       DII$PS  067042  DII$MS  067050  DII$IS  067054
                       DII$SS  067056  $DVI    067056  ILR$    067166
                       $ILR    067172  ABS     070516  STK$L   070534
                       STK$I   070540  STK$F   070544  LCI$    070554
                       LCO$    070622  MOF$RS  070662  MOF$RM  070670
                       MOF$RA  070700  MOF$RP  070704  MOI$RS  070710
                       MOL$RS  070710  MOI$RM  070714  MOI$RP  070720
                       MOI$RA  070722
SYS$I   070726 000030   (RW,I,LCL,REL,CON)
                       IPEEK   070726  IPOKE   070736  ITTINR  070746
USER$I  070756 000000   (RW,I,LCL,REL,CON)
$CODE   070756 007062   (RW,I,LCL,REL,CON)
                       SETUP @ 070756  AMPQND@ 072456  BASLIN  072760
                       INTGRT@ 073074  TRAP    073444  AMPLIT  074076
                       GETVAL  074304  LAB1    076300  LAB2    076346
                       LAB3    076414  LAB4    076672  INITCU  076740
                       ERACUR  077130  DRWCUR  077254  AMPDAT@ 077632
                       AMPCUR@ 100002
OTS$O   100040 000000   (RW,I,LCL,REL,CON)
SYS$O   100040 000000   (RW,I,LCL,REL,CON)
$DATAP  100040 002622   (RW,D,LCL,REL,CON)
```

```
OTS$D   102662 000000   (RW,D,LCL,REL,CON)
OTS$S   102662 000000   (RW,D,LCL,REL,CON)
SYS$S   102662 000004   (RW,D,LCL,REL,CON)
                       $SYSLB  102662  $LOCK   102664  $CRASH  102665
$DATA   102666 001100   (RW,D,LCL,REL,CON)
USER$D  103766 000000   (RW,D,LCL,REL,CON)
Segment size = 014724 = 3306.  words

OTS$I   067042 002414   (RW,I,LCL,REL,CON)
                       CLOSE   067042  $GETFI  067066  $$OPCL  067124
                       $$$ERR  067236  $$$DIS  067260  $OSTMI  067362
                       $OSTM   067366  IRR$    070516  $IRR    070522
                       IRW$    070546  $IRW    070552  $GETIN  071126
                       $SETIN  071164  DEF$    071272  $DEF    071276
                       SAVR4$  071372
SYS$I   071456 000000   (RW,I,LCL,REL,CON)
USER$I  071456 000000   (RW,I,LCL,REL,CON)
$CODE   071456 006206   (RW,I,LCL,REL,CON)
                       PLTAIB@ 071456  GOTOMA@ 071730  ERABUF@ 072024
                       INCEXT@ 072310  INCR    072374  MV2AI @ 073000
                       DATIN @ 073036  INP     073546  DATOUT@ 074230
                       OUP     074740  DIFAB @ 075422  ERAAIB@ 075560
                       TYPAIB@ 075654  WFCR  @ 076014  OUTPRO@ 076072
                       INPROG@ 076274  AIBIN @ 076476  AIBOUT@ 077214
OTS$O   077664 000000   (RW,I,LCL,REL,CON)
SYS$O   077664 000000   (RW,I,LCL,REL,CON)
$DATAP  077664 000634   (RW,D,LCL,REL,CON)
OTS$D   100520 000044   (RW,D,LCL,REL,CON)
OTS$S   100564 000002   (RW,D,LCL,REL,CON)
SYS$S   100566 000000   (RW,D,LCL,REL,CON)
$DATA   100566 000300   (RW,D,LCL,REL,CON)
USER$D  101066 000000   (RW,D,LCL,REL,CON)
        101066 000220   (RW,I,LCL,REL,CON)
                       LASER @ 101066  MOVBUR@ 101146  SHUTTR@ 101226
                       JOLTS @ 101252
Segment size = 012244 = 2642.  words
```

H. Typical cross reference table

```
******************** CROSS REFERENCE TABLE FOR MULTIP.MIZ ********************

ROUTINE (OVERLAY SEGMENT)       CALLS           ROUTINE(S) (OVERLAY SEGMENT)
--------------------------------------------------------------------------
ACQSCF (SEG1)                                     CIF (SEG1),CVTSCF (SEG1)
ACUMMP (SEG1)                                     CIF (SEG1),CVTSCF (SEG1)
ACUMSP (SEG1)                                     CIF (SEG1),CVTSCF (SEG1)

AIBIN  (SEG5)                                     INCEXT (SEG5)
AIBOUT (SEG5)                                     INCEXT (SEG5)
AMPCUR (SEG4)                                     AMPLIT (SEG4)

AMPDAT (SEG4)                                     AMPLIT (SEG4)
AMPLIT (SEG4)                                     GETVAL (SEG4)
AMPQND (SEG4)                                     SCANER (ROOT),BASLIN (SEG4)
```

```
CVTSCF (SEG1)                    *** NONE ***
DATIN  (SEG5)                    INCEXT (SEG5)
DATOUT (SEG5)                    INCEXT (SEG5)

DIFAB  (SEG5)                    *** NONE ***
ERAAIB (SEG5)                    *** NONE ***
ERABUF (SEG5)                    *** NONE ***

EXECUT (ROOT)                    OVERLAID EXECUTION MODULES
GETNAM (ROOT)                    NAMCLN (ROOT)
GETVAL (SEG4)                    WFMDP (ROOT), INITCU (ROOT)
                                 LAB1 (ROOT), LAB2 (ROOT),
                                 LAB4 (ROOT), ERACUR (ROOT)
                                 DRWCUR (ROOT), TYPMDP (ROOT)

GOTOMA (SEG5)                    *** NONE ***
INCEXT (SEG5)                    *** NONE ***
INPROG (SEG5)                    GETNAM (ROOT)

INTGRT (SEG4)                    GETVAL (SEG4),TRAP (SEG4)
JOLTS  (SEG5)                    *** NONE ***
LASER  (SEG5)                    *** NONE ***

LISTPR (SEG?)                    *** NONE ***
MAKPLT (ROOT)                    PICTUR (ROOT), AXES (ROOT),
                                 SCANER (ROOT), PLOTER (ROOT)
                                 LABMDP (ROOT), DRWMDP (ROOT)
                                 TYPMDP (ROOT), SETMDP (ROOT)
                                 ERATXT (ROOT), GOSFOR (ROOT)
MOVBUR (SEG5)                    *** NONE ***

MULTIP (ROOT)                    SETUP (SEG4), PROGRM (SEG2),
                                 GOTOMA (SEG5), EXECUT (ROOT),
                                 OUTPRO (SEG5), INPROG (SEG5),
                                 LISTPR (SEG?)
MV2AI  (SEG5)                    *** NONE ***
NAMCLN (ROOT)                    *** NONE ***

OUTPRO (SEG5)                    GETNAM (ROOT)
PLTAIB (SEG5)                    SETMDP (ROOT), AXES (ROOT),
                                 SCANER (ROOT), PLOTER (ROOT)
                                 LABMDP (ROOT)
PROGRM (SEG2)                    GETBUF (SEG2), GETFIL (SEG2)

SETUP  (SEG4)                    GETNAM (ROOT)
SHUTTR (SEG5)                    *** NONE ***
TYPAIB (SEG5)                    *** NONE ***
WFCR   (SEG5)                    *** NONE ***


****************** END OF CROSS REFERENCE TABLE ****************************
```

DISTRIBUTION LIST

No of Copies	Organization
12	Administrator Defense Technical Info Center ATTN: DTIC-DDA Cameron Station Alexandria, VA 22314
1	HQ DA DAMA-ART-M Washington, DC 20310
1	Commander US Army Materiel Cmd. ATTN: AMCDRA-ST 5001 Eisenhower Avenue Alexandria, VA 22333
1	Commander Armament R&D Center USA AMCCOM ATTN: SMCAR-TDC Dover, NJ 07801-5001
2	Commander Armament R&D Center USA AMCCOM ATTN: SMCAR-TSS Dover, NJ 07801-5001
1	Commander US Army Armament, Munitions and Chemical Cmd ATTN: SMCAR-ESP-L Rock Island, IL 61299
1	Director Benet Weapon Laboratory Armament R&D Center USA AMCCOM ATTN: SMCAR-LCB-TL Watervliet, NY 12189
1	Commander USA Aviation Rsch and Development Cmd ATTN: AMSAV-E 4300 Goodfellow Blvd. St. Louis, MO 63120
1	Director USA Air Mobility Rsch and Development Lab. Ames Research Center Moffett Field, CA 94035
4	Commander USA Research Officer ATTN: R. Ghirardelli D. Mann R. Singleton R. Shaw Research Triangle Park NC 27709
1	Commander U.S. Army Communications - Electronics Command ATTN: AMSEL-ED Fort Monmouth, NJ 07703
1	Commander U.S. Army Communications - Electronics Command (CECOM) CECOM R&D Technical Library ATTN: ASMEL-IM-L B 2700 Fort Monmouth, NJ 07703-5000
2	Commander USA AMCCOM ATTN: DRSMC-LCA-G D.S. Downs J.A. Lannon Dover, NJ 07801
1	Commander USA AMCCOM ATTN: DRSMC-LC, L. Harris Dover, NJ 07801
1	Commander USA AMCCOM ATTN: DRSMC-SCA-T L. Stiefel Dover, NJ 07801

DISTRIBUTION LIST

No of Copies	Organization
1	Commander USA Missile Command Research, Development, Engineering Center ATTN: AMSMI-RD Redstone Arsenal, AL 35898
1	Commander USA Missile & Space Command Intelligence Center ATTN: AIAMS-YDL Redstone Arsenal, AL 35898
2	Commander USA Missile Command ATTN: DRSMI-RK, D.J. Ifshin Redstone Arsenal, AL 35898
1	Commander USA Tank Automotive Cmd ATTN: AMSTA-TSL Warren, MI 48397-5000
1	Director USA TRADOC Systems Analysis ATTN: ATAA-SL WSMR, NM 88002
2	Commandant USA Infantry School ATTN: ATSH-CD-CSO-OR Fort Benning, GA 31905
1	Commander USA Development and Employment Agency ATTN: MODE-TED-SAB Fort Lewis, WA 98433
1	Office of Naval Rsch Department of the Navy ATTN: R.S. Miller, Code 432 800 N. Quincy Street Arlington, VA 22217

No. of Copies	Organization
1	Navy Strategic Systems Project Office ATTN: R.D. Kinert, SP 2721 Washington, DC 20376
1	Commander Naval Air Systems Cmd ATTN: J. Ramnarace, AIR-54111C Washington, DC 20360
3	Commander Naval Ordnance Station ATTN: C. Irish S. Mitchell P.L. Stang, Code 515 Indian Head, MD 20640
1	Commander Naval Surface Weapons Center ATTN: J.L. East, Jr., G-20 Dahlgren, VA 22448
2	Commander Naval Surface Weapons Center ATTN: R. Bernecker, R-13 G.B. Wilmot, R-16 Silver Spring, MD 20910
4	Commander Naval Weapons Center ATTN: R.L. Derr, Code 389 China Lake, CA 93555
2	Commander Naval Weapons Center ATTN: Code 3891, T. Boggs K.J. Graham China Lake, CA 93555
5	Commander Naval Research Lab, Code 6110 ATTN: L. Harvey J. McDonald E. Oran J. Shnur R.J. Doyle, Washington, DC 20375

DISTRIBUTION LIST

No of Copies	Organization
1	Commanding Officer Naval Underwater System Center Weapons Dept. ATTN: R.S. Lazar/Code 36301 Newport, RI 02840
1	Superintendent Naval Postgraduate School Dept. of Aeronautics ATTN: D.W. Netzer Monterey, CA 93940
6	AFRPL (DRSC) ATTN: R. Geisler D. George B. Goshgarian J. Levine W. Roe D. Weaver Edwards AFB, CA 93523
1	Air Force Armament Laboratory ATTN: AFATL/DLODL ATTN: O.K. Heiney Eglin AFB, FL 32542-5000
2	AFOSR ATTN: L.H. Caveny J.M. Tishkoff Bolling Air Force Base Washington, DC 20332
1	AFWL/SUL Kirtland AFB, NM 87117
1	NASA Langley Research Center ATTN: G.B. Northam/MS 168 Hampton, VA 23365
4	National Bureau of Standards ATTN: J. Hastie M. Jacox T. Kashiwagi H. Semerjian US Dept. of Commerce Washington, DC 20234
1	Aerojet Solid Propulsion Company ATTN: P. Micheli Sacramento, CA 95813
1	Applied Combustion Technology, Inc. ATTN: A.M. Varney P.O. Box 17885 Orlando, FL 32860
2	Atlantic Research Corp. ATTN: M.K. King 5390 Cherokee Avenue Alexandria, VA 22314
1	Atlantic Research Corp. ATTN: R.H.W. Waesche 7511 Wellington Road Gainesville, VA 22065
1	AVCO Everett Research laboratory Division ATTN: D. Stickler 2385 Revere Beach Parkway Everett, MA 02149
1	Battelle Memorial Institute Tactical Technology Center ATTN: J. Huggins 505 King Avenue Columbus, OH 43201
2	Exxon Research & Engineering Company ATTN: A. Dean M. Chou P.O. Box 45 Linden, NJ 07036
1	Ford Aerospace and Communications Corp. DIVAD Division Div. Hq., Irvine ATTN: D. Williams Main Street & Ford Road Newport Beach, CA 92663

DISTRIBUTION LIST

No of Copies	Organization
1	General Electric Armament & Electrical Systems ATTN: M.J. Bulman Lakeside Avenue Burlington, VT 05402
1	General Electric Co. ATTN: M. Lapp Schenectady, NY 12301
1	General Electric Ordnance Systems ATTN: J. Mandzy 100 Plastics Avenue Pittsfield, MA 01203
1	General Motors Rsch Lab Physics Department ATTN: R. Teets Warren, MI 48090
3	Hercules, Inc. Alleghany Ballistics Lab. ATTN: R.R. Miller P.O. Box 210 Cumberland, MD 21501
3	Hercules, Inc. Bacchus Works ATTN: K.P. McCarty P.O. Box 98 Magna, UT 84044
1	Hercules, Inc. AFATL/DLDL ATTN: R.L. Simmons Eglin AFB, FL 32542
1	Honeywell, Inc. Defense Systems Div. ATTN: D.E. Broden/ MS MN50-2000 600 2nd Street NE Hopkins, MN 55343
1	IBM Corporation ATTN: A.C. Tam Research Division 5600 Cottle Road San Jose, CA 95193
1	Director Lawrence Livermore National Laboratory ATTN: C. Westbrook Livermore, CA 94550
1	Lockheed Missiles & Space Company ATTN: George Lo 3251 Hanover Street Dept. 52-35/B204/2 Palo Alto, CA 94304
1	Los Alamos National Lab ATTN: B. Nichols T7, MS-B284 P.O. Box 1663 Los Alamos, NM 87545
1	Olin Corporation Smokeless Powder Operations ATTN: R.L. Cook P.O. Box 222 St. Marks, FL 32355
1	Paul Gough Associates ATTN: P.S. Gough 1048 South Street Portsmouth, NH 03801
2	Princeton Combustion Research Labs, Inc. ATTN: M. Summerfield N.A. Messina 475 US Highway One Monmouth Junction, NJ 08852
1	Hughes Aircraft Company ATTN: T.E. Ward 8433 Fallbrook Avenue Canoga Park, CA 91303

DISTRIBUTION LIST

No of Copies	Organization
1	Rockwell International Corporation Rocketdyne Division ATTN: J.E. Flanagan/HB02 6633 Canoga Avenue Canoga Park, CA 91304
3	Sandia National Labs. Combustion Science Dept ATTN: R. Cattolica D. Stephenson P. Mattern Livermore, CA 94550
1	Sandia National Labs. ATTN: M. Smooke Division 8353 Livermore, CA 94550
1	Science Applications, Inc. ATTN: R.B. Edelman 23146 Cumorah Crest Woodland Hills, CA 91364
2	Univ. of California, Santa Barbara Quantum Institute ATTN: K. Schofield M. Steinberg Santa Barbara, CA 93106
1	University of Southern California Dept. of Chemistry ATTN: S. Benson Los Angeles, CA 90007
1	Case Western Reserve University Division of Aerospace Science ATTN: J. Tien Cleveland, OH 44135
1	Cornell University Department of Chemistry ATTN: E. Grant Baker Laboratory Ithaca, NY 14853
1	University of Dayton Research Institute ATTN: D. Campbell AFRPL/PAP Stop 24 Edwards AFB, CA 93523
1	University of Florida Dept. of Chemistry ATTN: J. Winefordner Gainesville, FL 32611
3	Georgia Inst. of Technology School of Aerospace Eng. ATTN: E. Price Atlanta, GA 30332
2	Georgia Institute of Technology School of Aerospace Eng. ATTN: W.C. Strahle B.T. Zinn Atlanta, GA 30332
1	University of Illinois Dept. of Mech. Eng. ATTN: H. Krier 144 MEB, 1206 W.Green Street Urbana, IL 61801
1	Johns Hopkins Univ./APL Chemical Propulsion Information Agency ATTN: T.W. Christian Johns Hopkins Road Laurel, MD 20707
1	University of Minnesota Dept. of Mechanical Eng. ATTN: E. Fletcher Minneapolis, MN 55455
4	Pennsylvania State University Applied Research Lab. ATTN: G.M. Faeth K.k. Kuo H. Palmer M. Micci University Park, PA 16802

DISTRIBUTION LIST

No of Copies	Organization
1	Polytechnic Institute of NY ATTN: S. Lederman Route 110 Farmingdale, NY 11735
2	Princeton University Forrestal Campus Library ATTN: K. Brezinsky I. Glassman P.O. Box 710 Princeton, NJ 08540
1	Princeton University MAE Dept. ATTN: F.A. Williams Princeton, NJ 08544
1	Science Applications, Inc. ATTN: H.S. Pergament 1100 State Road, Bldg. N Princeton, NJ 08540
1	Space Sciences, Inc. ATTN: M. Farber Monrovia, CA 91016
4	SRI International ATTN: S. Barker D. Crosley D. Golden Tech. Lib 333 Ravenwood Avenue Menlo Park, CA 94025
1	Stevens Institute of Technology Davidson Laboratory ATTN: R. McAlevy, III Hoboken, NJ 07030
1	Teledyne McCormack-Selph ATTN: C. Leveritt 3601 Union Road Hollister, CA 95023
10	Central Intelligence Agency Offfice of Central Reference Dissemination Branch Room GE-47 HQS Washington, D.C. 20502
1	Thiokol Corporation Elkton Division ATTN: W.N. Brundige P.O. Box 241 Elkton, MD 21921
3	Thiokol Corporation Huntsville Division ATTN: D.A. Flanagan Huntsville, AL 35807
3	Thiokol Corporation Wasatch Division ATTN: J.A. Peterson P.O. Box 524 Brigham City, UT 84302
1	United Technologies ATTN: A.C. Eckbreth East Hartford, CT 06108
2	United Technologies Corp. ATTN: R.S. Brown R.O. McLaren P.O. Box 358 Sunnyvale, CA 94088
1	Universal Propulsion Company ATTN: H.J. McSpadden Black Canyon Stage 1 Box 1140 Phoenix, AX 85029
1	Veritay Technology, Inc. ATTN: E.B. Fisher P.O. Box 22 Bowmansville, NY 14026
1	Brigham Young Univ. Dept. of Chemical Eng. ATTN: M.W. Beckstead Provo, UT 84601
1	California Institue of Technology Jet Propulsion Lab. ATTN: MS 125/159 4800 Oak Grove Drive Pasadena, CA 91103

DISTRIBUTION LIST

No of Copies	Organization
1	California Institue of Technology ATTN: F.E.C. Culick/ MC 301-46 204 Karman Lab. Pasadena, CA 91125
1	Univ. of California, Berkeley Mechanical Engineering Department ATTN: J. Daily Berkeley, CA 94720
1	Univ. of California Los Alamos National Laboratory ATTN: T.D. Butler P.O. Box 1663, Mail Stop B216 Los Alamos, NM 87545
2	Purdue University School of Aeronautics and Astronautics ATTN: R. Glick J.R. Osborn Grissom Hall West Lafayette, IN 47907
3	Purdue University School of Mechanical Engineering ATTN: N.M. Laurendeau S.N.B. Murthy D. Sweeney TSPC Chaffee Hall West Lafayette, IN 47906
1	Rensselaer Polytechnic Institute Dept. of Chemical Engineering ATTN: A. Fontijn Troy, NY 12181
2	Southwest Research Institute ATTN: R.E. White A.B. Wenzel 8500 Culebra Road San Antonio, TX 78228
1	Stanford University Dept. of Mechanical Engineering ATTN: R. Hanson Stanford, CA 93106
1	University of Texas Dept. of Chemistry ATTN: W. Gardiner Austin, TX 78712
1	University of Utah Dept. of Chemical Engineering ATTN: G. Flandro Salt Lake City, UT 84112
1	Virginia Polytechnic Institute & State University ATTN: J.A. Schetz Blackburg, VA 24061

Aberdeen Proving Ground

Dir, USAMSAA
ATTN: AMXSY-D
AMXSY-MP, H. Cohen
Cdr, USATECOM
ATTN: AMSTE-TO-F
Cdr, CRDC, AMCCOM
ATTN: SMCCR-RSP-A
SMCCR-MU
SMCCR-SPS-IL

USER EVALUATION SHEET/CHANGE OF ADDRESS

This Laboratory undertakes a continuing effort to improve the quality of the reports it publishes. Your comments/answers to the items/questions below will aid us in our efforts.

1. BRL Report Number _____ Date of Report _____

2. Date Report Received _____

3. Does this report satisfy a need? (Comment on purpose, related project, or other area of interest for which the report will be used.) _____

_____

_____

4. How specifically, is the report being used? (Information source, design data, procedure, source of ideas, etc.) _____

_____

_____

5. Has the information in this report led to any quantitative savings as far as man-hours or dollars saved, operating costs avoided or efficiencies achieved, etc? If so, please elaborate. _____

_____

_____

6. General Comments. What do you think should be changed to improve future reports? (Indicate changes to organization, technical content, format, etc.)

_____

_____

_____

CURRENT ADDRESS

_____
Name

_____
Organization

_____
Address

_____
City, State, Zip

7. If indicating a Change of Address or Address Correction, please provide the New or Correct Address in Block 6 above and the Old or Incorrect address below.

OLD ADDRESS

_____
Name

_____
Organization

_____
Address

_____
City, State, Zip

(Remove this sheet along the perforation, fold as indicated, staple or tape closed, and mail.)

FOLD HERE

Director
U.S. Army Ballistic Research Laboratory
ATTN: SLCBR-DD-T
Aberdeen Proving Ground, MD 21005-5066

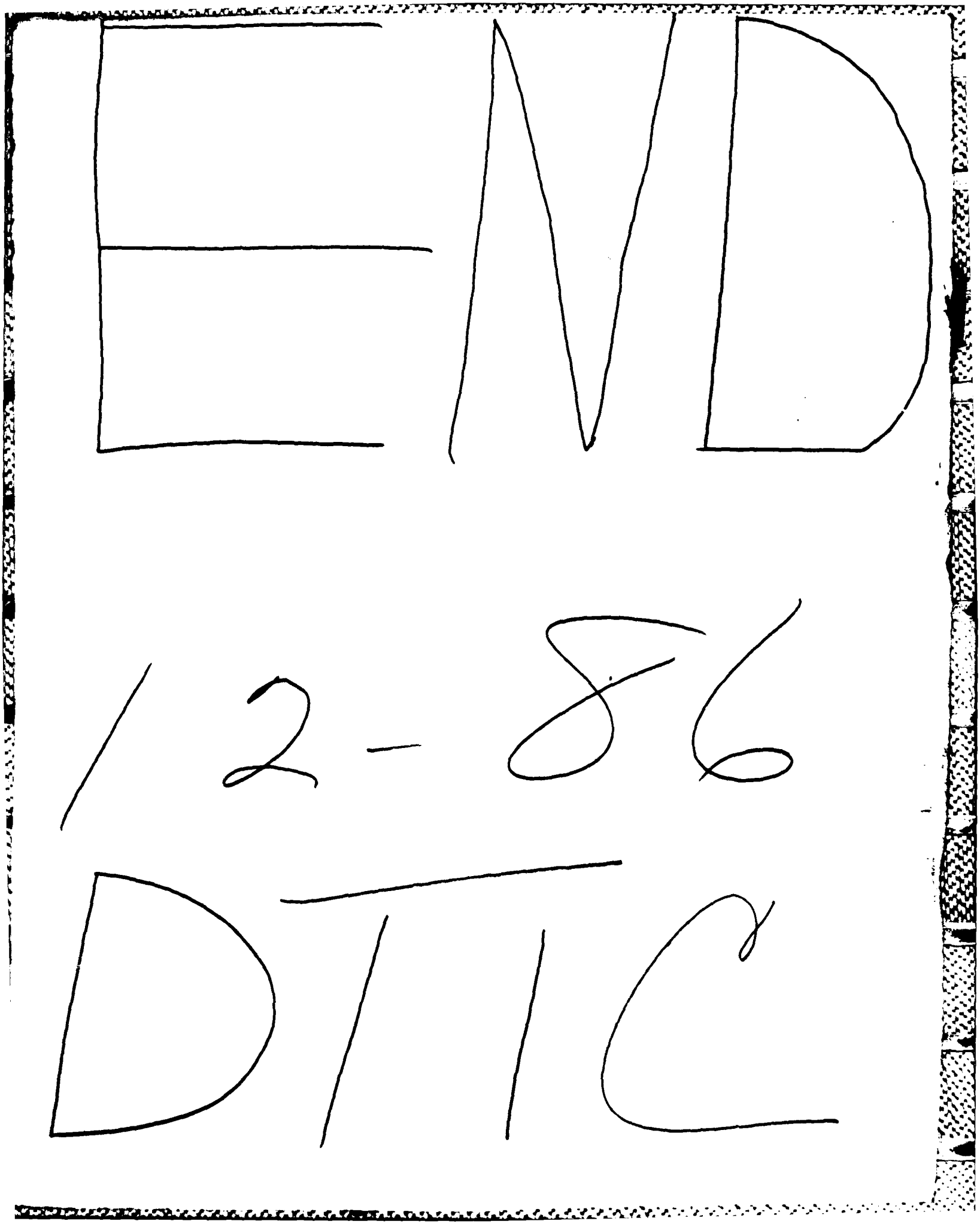
NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

OFFICIAL BUSINESS
PENALTY FOR PRIVATE USE, $300

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO 12062 WASHINGTON, DC
POSTAGE WILL BE PAID BY DEPARTMENT OF THE ARMY

Director
U.S. Army Ballistic Research Laboratory
ATTN: SLCBR-DD-T
Aberdeen Proving Ground, MD 21005-9989

FOLD HERE


END
12-86
DTIC